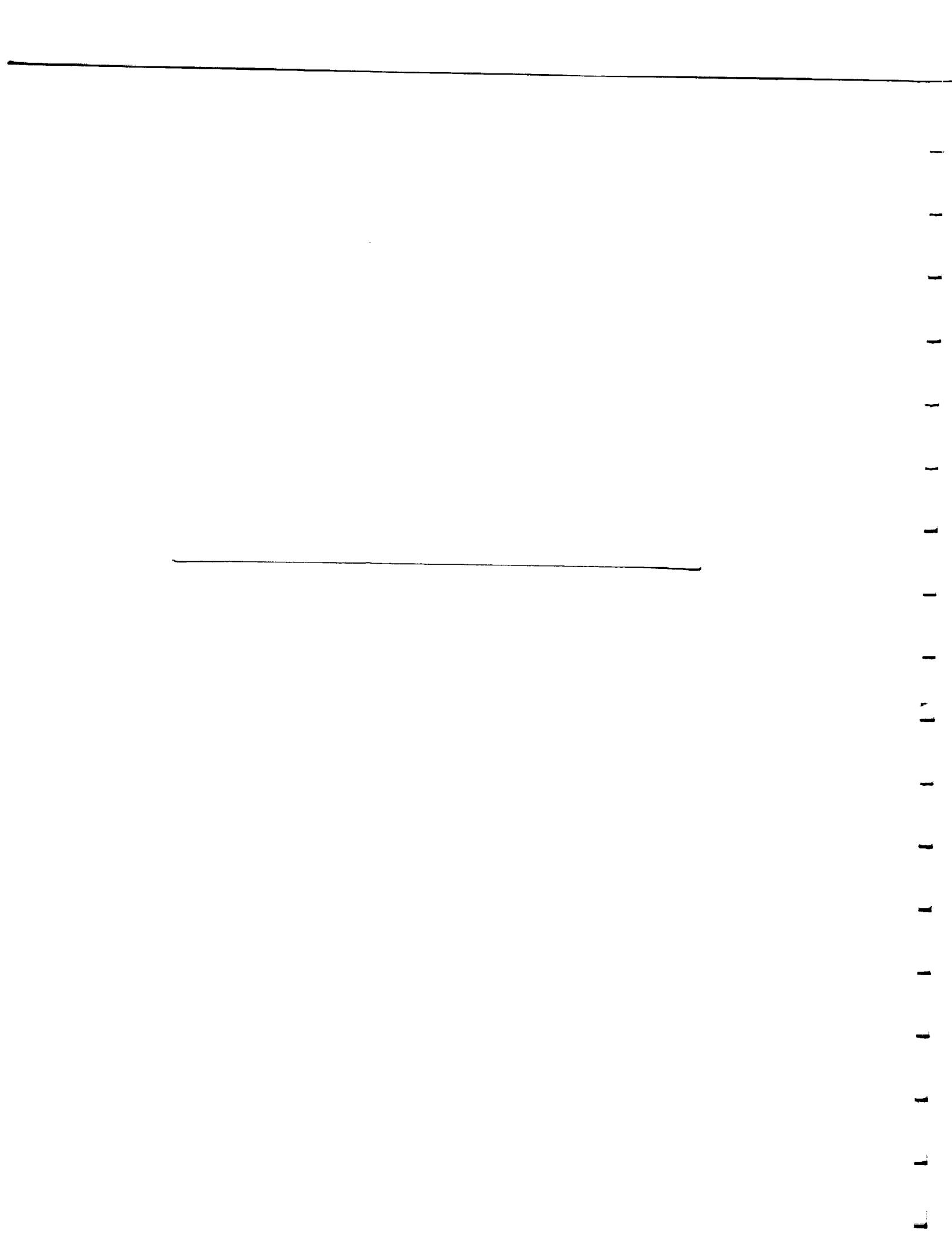


NAG5-1491



THE KLIPSCH SCHOOL OF
ELECTRICAL AND COMPUTER
ENGINEERING

TECHNICAL REPORT SERIES



**CARRIER ESTIMATION USING CLASSIC SPECTRAL
ESTIMATION TECHNIQUES FOR THE PROPOSED
DEMAND ASSIGNMENT MULTIPLE ACCESS SERVICE**

Bradley James Scaife, B.S.

NMSU-ECE-99-007 August 1999

CARRIER ESTIMATION USING CLASSIC SPECTRAL ESTIMATION
TECHNIQUES FOR THE PROPOSED DEMAND ASSIGNMENT
MULTIPLE ACCESS SERVICE

BY
BRADLEY JAMES SCAIFE, B.S.

A Thesis submitted to the Graduate School
in partial fulfillment of the requirements
for the Degree
Master of Science in Electrical Engineering
Major Subject: Electrical Engineering

New Mexico State University

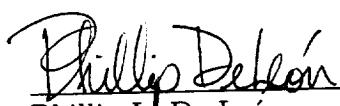
Las Cruces, New Mexico

August 1999

"Carrier Estimation Using Classic Spectral Estimation Techniques for the Proposed Demand Assignment Multiple Access Service," a thesis prepared by Bradley James Scaife in partial fulfillment of the requirements for the degree, Master of Science, has been approved and accepted by the following:



Timothy J. Pettibone
Dean of the Graduate School



Phillip L. De León
Chair of the Examining Committee

July 26, 1999
Date

Committee in charge:

Dr. Phillip L. De León, Chair

Dr. Gerald J. Dunn

Dr. Stephen Horan

Dr. James P. Le Blanc

ACKNOWLEDGMENTS

The author would like to take the opportunity to thank Frank Hartman, Cliff Baxtor, and the entire White Sands Complex Staff for their support and for the opportunity to gather test data in a very busy environment. Additionally, the author wishes to express gratitude to Lawrence Alvarez for invaluable assistance on numerous occasions in setting up hardware and test beds.

The author would like to thank Dr. Paul Klipsch for his incredible service to New Mexico State University, the field of electrical engineering, and to the author personally. It was an honor to be a graduate scholarship recipient but a greater honor to meet Dr. Klipsch personally.

The author would like to express his appreciation and admiration for the professors of the Klipsch School of Electrical Engineering and in particular to Dr. James Le Blanc for excellent instruction and invaluable assistance with this work.

The author would like to take the opportunity to express a lifetime of gratitude for the many lessons taught him by his advisor Dr. Phillip L. De León. His work on this project and support of it as well as his willingness to direct, correct, and instruct with seemingly endless patience has made him more friend than advisor.

A final word of gratitude to my wife Nancy for ever demonstrating faith, hope, and love, but the greatest of these....

VITA

February 24, 1969—Born in Medford, Oregon

1987—Graduated from Ashland High School, Ashland Oregon

1994—Graduated from Oregon Institute of Technology, Klamath Falls, Oregon

1994-1997—Intel Corporation, Portland, Oregon

1998-1999—Research Assistant, New Mexico State University Manuel Lujan
Center for Space Telemetering

1999—Teaching Assistant, Klipsch School of Electrical and Computer
Engineering, New Mexico State University

PROFESSIONAL SOCIETIES

Institute of Electrical and Electrical Engineers

PUBLICATIONS

Scaife, Bradley J. 1998. Doppler Shifted Spread Spectrum Carrier Recovery Using
Real-Time DSP Techniques. *International Telemetering Conference.*

FIELD OF STUDY

Major Field: Signal Processing and Computer Architecture

ABSTRACT

CARRIER ESTIMATION USING CLASSIC SPECTRAL ESTIMATION
TECHNIQUES FOR THE PROPOSED DEMAND ASSIGNMENT
MULTIPLE ACCESS SERVICE

BY

BRADLEY JAMES SCAIFE, B.S.

Master of Science in Electrical Engineering

New Mexico State University

Las Cruces, New Mexico, 1999

Dr. Phillip L. De León, Chair

In any satellite communication, the Doppler shift associated with the satellite's position and velocity must be calculated in order to determine the carrier frequency. If the satellite state vector is unknown then some estimate must be formed of the Doppler-shifted carrier frequency. One elementary technique is to examine the signal spectrum and base the estimate on the dominant spectral component. If, however, the carrier is spread (as in most satellite communications) this technique may fail unless the chip rate-to-data rate ratio (processing gain) associated with the carrier is small.

In this case, there may be enough spectral energy to allow peak detection against a noise background.

In this thesis, we present a method to estimate the frequency (without knowledge of the Doppler shift) of a spread-spectrum carrier assuming a small processing gain and binary-phase shift keying (BPSK) modulation. Our method relies on an averaged discrete Fourier transform along with peak detection on spectral match filtered data. We provide theory and simulation results indicating the accuracy of this method. In addition, we will describe an all-digital hardware design based around a Motorola DSP56303 and high-speed A/D which implements this technique in real-time. The hardware design is to be used in NMSU's implementation of NASA's demand assignment, multiple access (DAMA) service.

Contents

List of Figures	ix
List of Tables	xi
Frequently Used Terminology	xii
1 Overview	1
1.1 Introduction	1
1.2 Comparison of Goddard and NMSU's DAMA Proposals	2
1.3 Proposed Solution	3
1.4 Simulations, Test Data, and Theory	4
1.5 DAMA Hardware	6
2 DAMA Project Description	8
2.1 Current WSC Operations	8
2.2 NMSU's Proposal	10
2.3 Spread Spectrum Fundamentals	12
2.4 Operating Parameters Description	16
2.5 Carrier Estimation Problem	18
2.6 Proposed Solution	19
3 Theory and Simulation	24
3.1 Signal Description	24
3.2 Estimation Accuracy Theory	25
3.3 Comparison of Theory to Simulation	29

3.4	Simulation Model Description	31
3.5	Simulation Results	32
3.6	Theoretical/Simulation Data Summary	38
4	WSC Data Collection Experiment	39
4.1	Motivation of WSC Data Collection Experiment	39
4.2	Data Collection Setup	40
4.3	WSC Collected Data Processing	43
4.4	WSC Data Results Compared to Simulation	44
4.5	Conclusions of WSC Data Collection	46
5	Carrier Estimation Hardware and Software	48
5.1	Motorola DSP56303EVM Description	48
5.2	Burr Brown 800kHz A/D	51
5.3	A/D Interface Board	51
5.4	Additional Hardware	52
5.5	Software Description for Real-Time Carrier Estimation	53
6	Conclusions and Future Work	56
A	Matlab Simulation Code	57
B	Motorola DSP 56303EVM Code	94
References		134

List of Figures

2.1	Illustration of Doppler Shift to DAMA Carrier	9
2.2	DAMA Carrier Placement Against MA Spectrum	12
2.3	TDRSS MA Spectrum at IF from Actual Data	13
2.4	2047 PN Code Implementation	14
2.5	Spreading Effect of Processing Gains	16
2.6	Flowchart for Carrier Estimation Algorithm	23
3.1	Signal Space Diagram for BPSK	25
3.2	Comparison of Theoretical Curve Against Simulation	30
3.3	Family of Curves for Non-SMF Case	33
3.4	Non-SMF Frequency Estimation for PG = 10, SNR = 2 dB	34
3.5	Non-SMF Frequency Estimation for PG = 20, SNR = 2 dB	34
3.6	Non-SMF Frequency Estimation for PG = 100, SNR = 2 dB	35
3.7	Family of Curves for SMF Case	35
3.8	SMF Frequency Estimation for PG = 10, SNR = 2 dB	36
3.9	SMF Frequency Estimation for PG = 40, SNR = 2 dB	37
3.10	SMF Frequency Estimation for PG = 100, SNR = 2 dB	37
4.1	Experiment Setup	40
4.2	Placement of Test Frequencies Near TDRSS Null	42
4.3	Comparison of Simulated Results vs. WSC Captured Data	45

4.4	Carrier Estimation Summary	47
5.1	DSP56300 Core System Block Diagram	49
5.2	Locking Tone Generation	53

List of Tables

1	Test Sets Captured	43
---	------------------------------	----

Frequently Used Terminology

- AWGN: Additive White Gaussian Noise
- BPSK: Binary Phase Shift Keying(ed)
- BW: Bandwidth
- CDMA: Code Division Multiple Access
- DAMA: Demand Access Multiple Assignment
- DSO: Digital Storage Oscilloscope
- DSP: Digital Signal Processing/Processor
- MA: Multiple Access
- PG: Processing Gain
- PN: Psuedo-Noise
- PSD: Power Spectral Density
- Rx: Receive
- SMF: Spectral Matched Filter
- SN: Space Network
- SNR: Signal to Noise Ratio

- SS: Spread Spectrum
- TDMA: Time Division Multiple Access
- TDRS: Tracking and Data Relay Satellite
- TDRSS: Tracking and Data Relay Satellite System
- Tx: Transmit
- WSC: NASA's White Sands Complex

1 Overview

1.1 Introduction

In an effort to provide increased access to NASA's Space Network (SN), a Demand Access Multiple Assignment communication scheme has been proposed. Under this scheme, users would have the option to communicate short information packets at low data rates *on demand*. This scheme is driven by the increased ability of modern satellites (spacecraft) to detect error conditions on board the satellite [1]. Under current SN operations, communication services are pre-scheduled and the schedules often have significant delay and are not easily modified. The DAMA service is to be designed such that it operates independently of the current scheduled Multiple Access (MA) service.

The SN consists of geostationary Tracking and Data Relay Satellites (TDRS) that operate as a virtual "radio frequency (RF) mirror" for data transmissions from/to low Earth orbiting (LEO) spacecraft communicating to/from ground stations. These LEO spacecraft are in orbit about the earth and due to their relative motion to a receiving TDRS, a Doppler shift is induced in transmissions. Since the MA service is pre-scheduled, these Doppler shifts may be accounted for allowing the ground station to detect and receive the LEO signal. In the case of the DAMA system, where data transmissions are to be scheduled on demand,

the Doppler shift information may be unknown and thus a system must be designed to estimate (to within ± 3 kHz) the carrier of the LEO signals.

1.2 Comparison of Goddard and NMSU's DAMA Proposals

Two independent proposals have been offered to implement the DAMA service. The first has been proposed by NASA/Goddard Space Flight Center (GSF) and the second by New Mexico State University. The Goddard proposal intends to provide continuous tracking of all LEO satellites equipped with DAMA capability. Thus with the state vectors of all of these LEO satellites known, the Doppler shift of the carrier can be computed much like for the MA service. Whereas the Goddard proposal must maintain these state vectors so that the ground station receiver may demodulate, the NMSU proposal forgoes LEO satellite state vector knowledge to simplify the required ground station hardware. In the NMSU proposal only a single element of the TDRS antenna array is used as a global beacon [1]. With a global beacon configuration, even a satellite that is experiencing alignment problems may transmit an emergency message to ground station users utilizing the DAMA communication system. As the state vector of the communicating spacecraft has been given up, the ground station will not know the position of transmitting LEO space vehicle and thus the Doppler shift cannot be accounted for [2]. Previous work has shown that the Doppler shift of a LEO spacecraft and

a TDRSS can vary by as much as ± 50 kHz which is outside of the ground station receiver (GSR) tolerance of ± 3 kHz [3].

The fundamental problem to NMSU's proposal lies with estimating the carrier of the transmitted signal to within the tolerance of the GSR. Hardware must be developed that will provide a locking tone, accurate to within the ground station tolerance, in order for the ground station to demodulate. The problem is exacerbated by the nature of the DAMA signal. The DAMA carrier is to employ a Spread Spectrum (SS) scheme. SS signals tend to suppress spectral peaks of carriers and have responses that are spread out over a wider bandwidth. As we shall show below, this makes the proposed solution to carrier estimation more difficult.

1.3 Proposed Solution

The proposed solution to the carrier estimation problem described above is to employ classical Digital Signal Processing (DSP) spectral estimation techniques to estimate the carrier frequency. We shall employ a Discrete Fourier Transform (DFT) to generate magnitude squared spectral data from the received signal. The resolution of the DFT will be set to provide accuracy to within the ground station tolerance. A single iteration of this process will not be sufficient to protect the carrier estimation from noise so we will average the magnitude squared data to limit the effects of noise. This process results in a *periodogram*. Having obtained the

periodogram of the received signal, we shall employ a frequency domain matched filter to maximize the Spectrum-to-Noise ratio (SPNR). The frequency domain matched filter will be predetermined based primarily upon the predefined SS frequency characteristics, such as processing gain (PG) and power, of the received signal. The results of the application of the matched filter, like those of the time domain equivalent, provide an optimal solution by enhancing the spectrum prior to searching. We then search the enhanced periodogram for a peak with which we base our carrier estimation. The accuracy of this technique will be shown to depend primarily on the PG of the DAMA carrier and the Signal-to-Noise ratio(SNR).

1.4 Simulations, Test Data, and Theory

At the core of the proof of concept for the proposed solution is a simulation model designed in Matlab. The simulation models the DAMA carrier against additive white Gaussian noise (AWGN), where we use AWGN to effectively model the white noise like spectrum of the MA service [2]. Having approximated the TDRSS channel by its most critical feature(presence of the MA service) we add the DAMA carrier and simulate carrier estimation based upon the approach described in section 1.3. The simulation model provides estimation accuracy as a function of the SNR and the PG of the DAMA carrier. We will show that accurate estimation,

to within the ground station tolerance, is achievable 80%-90% of the time for the given DAMA data rates and corresponding spreading rates.

To verify the accuracy of the simulation model, we include the results of an experiment with test data gathered at NASA's White Sands Complex (WSC). In this experiment, actual data vectors were streamed to a ground station transmitter for transmission to a TDRS (in orbit) and sent back to the ground station receiver. The parameters of the experiment were set such that we could observe several key issues with carrier estimation. The data gathered was processed with the proposed algorithm and compared to simulation results. The most significant conclusion of the experiment was that carrier estimation with the collected data was nearly equivalent to results obtained through simulation. The simulation is then recognized to accurately model the actual TDRSS channel.

We have developed a theoretical analysis that leads to a rough approximation for carrier estimation accuracy. The analysis of carrier frequency estimation is based upon use of the DFT, and the description of carrier estimation accuracy as a random variable [4]. From this description and the use of various approximations, we obtain an expression that describes the root mean square error (RMSE) between the actual carrier frequency and the estimate. The result is expressed as a function of SNR, data (chip) rate, and window type and length. Though the approximations break down in low SNR cases, in the higher SNR cases theory agrees with simulation results.

1.5 DAMA Hardware

As described above, we seek to provide the ground station with an accurate carrier frequency based upon our estimation. We perform this estimation through the use of specifically designed hardware. The base of the hardware utilizes Motorola's DSP56303EVM (EVM). The EVM utilizes Motorola's DSP56300 core which is capable of 80 million instructions per second at 80 MHz and has enough available on-chip memory to implement the algorithm described above. As we will be required to sample the incoming signals at rates greater than that allowed by the EVM, we have integrated an 800 kHz 12 bit Burr-Brown ADS7810/19 analog to digital converter (A/D) into the design. To interface the EVM and the A/D requires some additional logic and level translators that are implemented on an additional interface card. The card allows the EVM to control the A/D while allowing samples from the A/D to be passed directly into the memory of the EVM for processing. These three components, excluding some additional analog pre-processing and post-processing equipment, make up the core of the carrier estimation hardware. The hardware is designed to receive signals that have been filtered and frequency shifted to baseband, estimate the carrier, and then provide a locking tone to the GSR. The GSR will use this carrier estimate to demodulate the DAMA carrier.

The hardware has been tested with synthesized waveforms as well as actual waveforms captured during the WSC experiment and performs as designed/required.

2 DAMA Project Description

2.1 Current WSC Operations

TDRSS was originally devised by NASA as an efficient means to control costs associated with providing a ground station for each satellite [5]. The concept of a space network was formed where users could transmit and receive all communications through a common ground station. NASA operates TDRSS as a space network (SN) using it to provide customers with communication access to their Low Earth Orbiting (LEO) spacecraft. The SN consists of six geostationary Tracking and Data Relay Satellites (TDRS) located 22,250 miles in orbit and a ground station located at the WSC (other operational ground stations exist as well) [5]. The function of a TDRS is to act as a virtual “RF mirror” through which communication signals are relayed between user spacecraft and the ground station. An antenna array, located on each TDRS, is tuned by weighting antenna elements to provide a spot beacon to the spacecraft. This requires a unique weighting vector and associated signal processing equipment for each user spacecraft [1]. Two communication schemes are used by the SN to fulfill various communication needs [5] [3]:

- multiple access (MA) at low data rates of 100 bps to 50 kbps operating in the S-band (2.1031 GHz - 2.1097 GHz forward service, 2.2845 GHz - 2.2905 GHz return) and using CDMA spread spectrum with a chip rate of 3 Mchips/s.

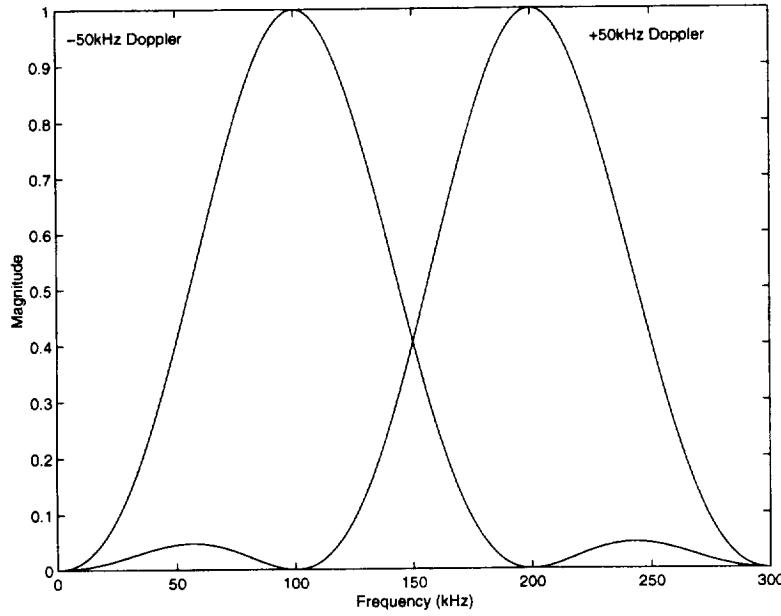


Figure 2.1: Illustration of Doppler Shift to DAMA Carrier

- single access (SA) at up to 300 kbps operating in either the S-band or the K-band (2.0204 GHz - 2.1233 GHz forward service, 2.2 GHz - 2.3GHz return) using TDMA.

The SN is able to provide 80% - 100% coverage for LEO spacecraft and is capable of simultaneously supporting 26 user spacecraft.

TDRSS consists of geostationary satellites but the LEO spacecraft that use this system are not necessarily geostationary. We know that signals originating from a source moving relative to a TDRS will experience a Doppler shift [6]. It has been shown that the Doppler shift of these signals can be as much as ± 50 kHz [3]. This is illustrated in Figure 2.1. The GSR normally maintains the state vector of

the satellite it is intended to communicate with and hence can simply calculate an estimate of the Doppler shift. Provided that the estimated Doppler shift is within ± 3 kHz of the actual Doppler shift, the ground station can demodulate the received signal. In any system where one would forgo knowledge of the state vector of these satellites, the result would be that in general the GSR could not synchronize to the Doppler shifted carrier and thus the carrier need be estimated.

The SN currently works under a scheduling process whereby a request for service must be made in advance (prescheduling) to utilize the SN. The scheduling delay often takes as much as 21 days for the request to be processed [3]. While the request can be serviced quicker in emergencies, the delay does not allow customers to react in near real-time to emergency situations that may arise with a user spacecraft. DAMA is a proposal that seeks to provide on demand communications between a user and their spacecraft without the need for prescheduling.

2.2 NMSU's Proposal

The initial scope of the NMSU proposal is towards implementing a "911" service where satellites that have gone into an error state may communicate this to the user when it detects such a condition. The eventual scope is to provide this service as a standard service to all DAMA capable spacecraft that require only low data rates with small data packets. Additionally it must be expanded to allow *multiple access*—or use by multiple users. For this thesis we assume a

single DAMA user at a time. The algorithm to be developed below is scalable to allow for multiple DAMA users at some point in the future.

The NASA GSFC proposal seeks to implement the DAMA service by maintaining the state vector information for each user spacecraft that is DAMA capable by continuously tracking each of these spacecraft. As in the MA service, with the state vector of the spacecraft known, it is routine to estimate the Doppler-shifted carrier and provide this estimate to the GSR. In contrast, NMSU's proposal gives up this state vector knowledge so that the DAMA ground station equipment is simplified. This leads to a problem with the Doppler estimation as it now must be estimated and supplied by means other than from the state vector. We propose a solution to this problem with the algorithm to be developed below that will execute on the hardware that was also developed to provide the GSR with this Doppler estimate. DAMA is to be implemented with a SS BPSK modulated communication scheme like the MA service described above. However, there are certain restrictions that determine the parameters of the scheme. The DAMA carrier is to be placed just inside the first upper TDRSS null as observed in Figure 2.2. The carrier will be placed such that a maximal Doppler shift of +50 kHz will not place the carrier too near the null so that the rolloff of the TDRSS channel and other associated GSR equipment, which bandpass filters on the mainlobe, will not adversely affect carrier estimation. The signal in Figure 2.3 consists of the MA service and the DAMA signal and demonstrates the overall response of

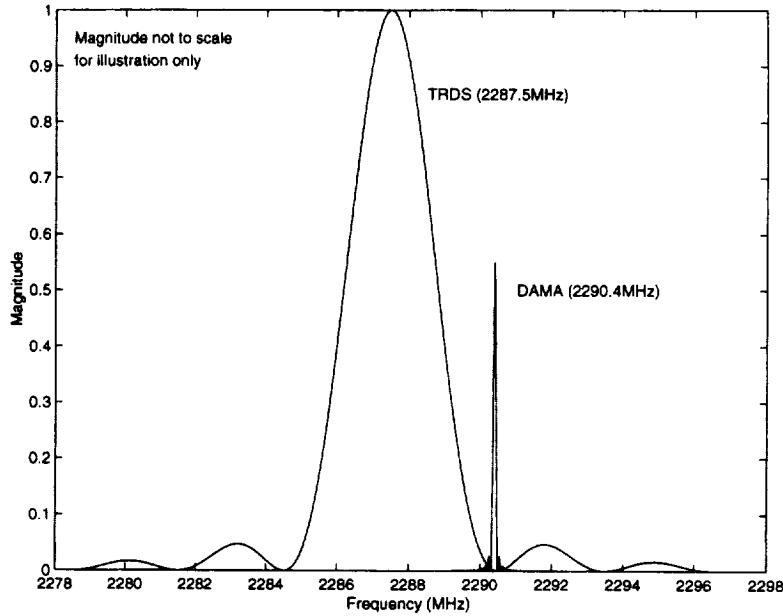


Figure 2.2: DAMA Carrier Placement Against MA Spectrum

the TDRSS system with the addition of the DAMA signal. We furthermore see the effects of sidelobe rejection of the TDRSS system.

2.3 Spread Spectrum Fundamentals

To discuss the operational parameters of the proposed DAMA carrier estimation, it will first be necessary to provide some fundamentals of SS communication schemes and definitions of important parameters. These parameters directly affect carrier estimation performance.

We begin with a basic and widely used definition for SS systems: SS systems are distinguished by the characteristic that their signals consume a bandwidth greater than the information rate [7]. Though there are several different tech-

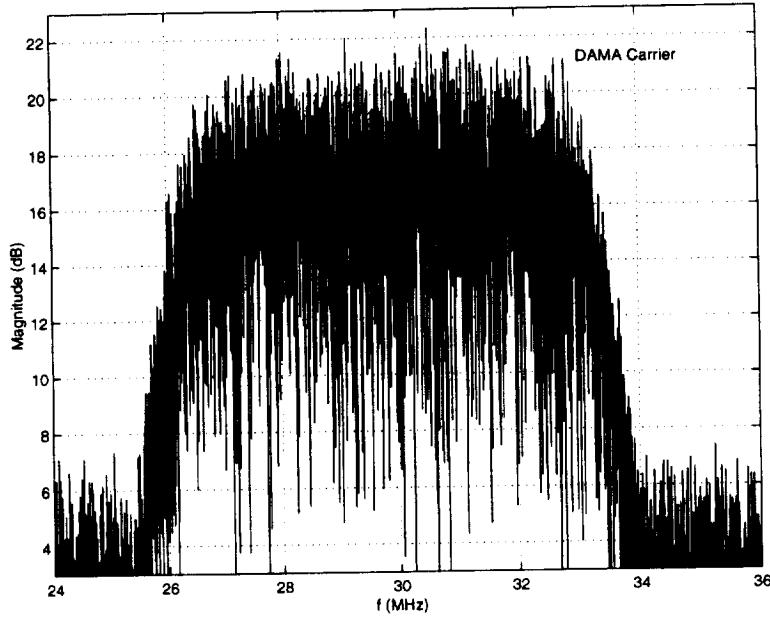


Figure 2.3: TDRSS MA Spectrum at IF from Actual Data

niques for implementing the spreading, we focus on the technique known as spread spectrum by direct sequence (DS). DS spread systems implement a scheme where the information data is acted upon by pseudo-noise (PN) data, whose elements are referred to as chips, to produce a spread spectrum bandwidth (BW). The ratio of chips to bits is typically an integer and the chip rate is often much higher than the data rate. This ratio is defined as the processing gain (PG) where

$$PG = \frac{R_c}{R_b} \quad (2.1)$$

and R_c is the chip rate in chips/s and R_b is the data rate in bits per second (bps). The PG also describes the ratio of chips/bit from which we see that each bit will be acted upon by PG chips through the use of *modulo-2 addition*. The PN code

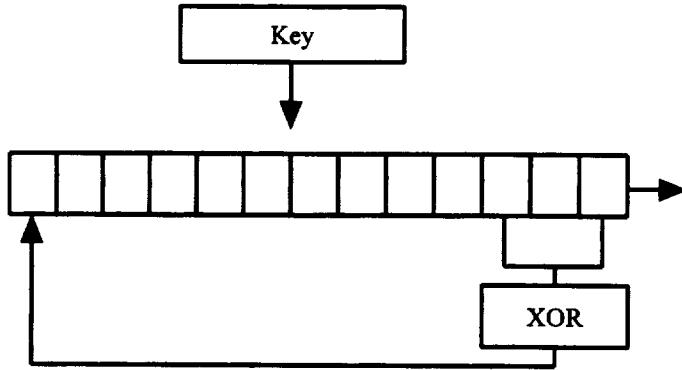


Figure 2.4: 2047 PN Code Implementation

sequence, \mathbf{C}_i , has the property that it approximates a white noise sequence and is periodic. \mathbf{C}_i is designed purposely such that

$$\mathbf{C}_i^T \mathbf{C}_j = \delta(i - j) \quad (2.2)$$

indicating orthogonality between PN codes of equal length but different “keys”.

In practice, PN codes are only approximately orthogonal. The number of 1’s and 0’s, with $\mathbf{C}_i \in \{0, 1\}$, differ by at most one. Many different techniques exist for the generation of these codes and we provide the 2047 PN code as an example. Though the 2047 PN code exhibits the qualities of white noise it is in fact periodic with period 2047. This PN code may be viewed as a primitive polynomial and implemented with a shift register as seen in Figure 2.4.

In general the initial state of the shift register is a “key” and each key represents a different PN code that is orthogonal to other PN codes as described in (2.2). In this manner each PN code operates as an orthogonal basis function for each vector

of data. Multiple users are allowed in the same bandwidth precisely because each message is orthogonal to the other.

For the MA service, $R_c = 3$ Mchips/s. From theory it is known that the BW consumed by this modulation scheme follows the relation:

$$BW \propto 2R_c \quad (2.3)$$

where BW is the bandwidth, and R_c is the chip or spreading rate [8]. From (2.3) we observe that the MA service will occupy approximately 6 MHz of BW as is illustrated in Figure 2.2 and Figure 2.3.

We will show that PG plays a large role in carrier estimation but first it is useful to see how PG will affect the BW of a signal. As we “spread” a carrier more and more (increase R_c relative to R_b), the spectrum of the carrier will tend to spread out and flatten. This can be observed in Figure 2.5 below where we have spread a BPSK at various rates. Both PG’s in Figure 2.5 are relatively low however, it will be shown that low PG’s are required for accurate carrier estimation. Estimation of the carrier becomes more difficult at higher PGs since the carrier power is not concentrated over a small band of frequencies, which would result in a sharp

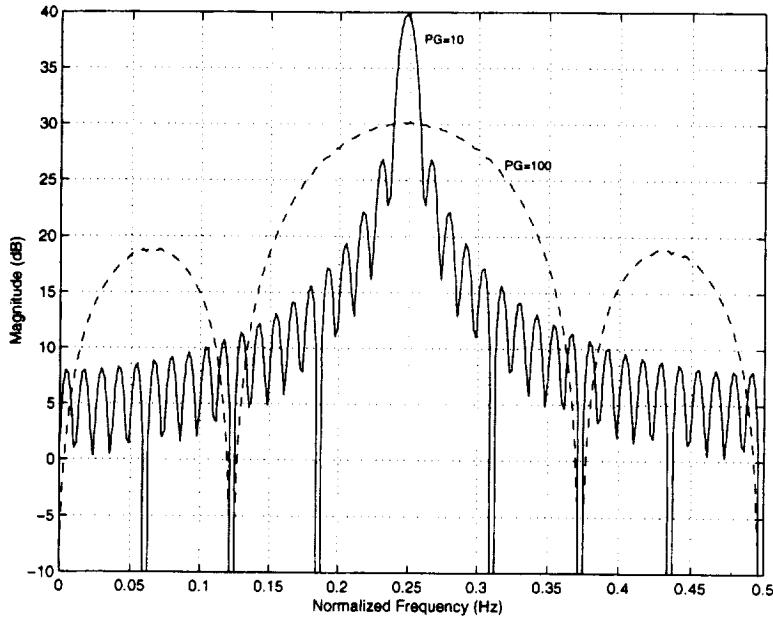


Figure 2.5: Spreading Effect of Processing Gains

spectral peak, but rather distributed over a wider range. The peak in the SS case has less power and therefore noise may bury it (as intended for SS systems).

2.4 Operating Parameters Description

We now describe some of the communication parameters of the DAMA proposal. We are particularly interested in three parameters when dealing with digital communication systems: power, bit rate, and probability of bit error. For DAMA carrier estimation, we are not required to demodulate the signal and thus do not look at the probability of bit error. The first two, however, will affect our ability to perform carrier estimation.

For the purposes of this thesis, we will describe DAMA power in two different ways. The first will be SNR in *dB* of the DAMA carrier-to-MA spectrum. We use this SNR definition when observing simulated work since the MA spectrum was modeled as AWGN. We can gain insight into this definition by observing Figure 2.3. The DAMA carrier will exhibit a peak against the *passband* of the MA service. Thus, we are interested in the DAMA power to that flat passband. The second way of describing power will be as the ratio C_b/N_o which is used in WSC operations. This is a measure of DAMA carrier power to the noise floor. Through observations of spectra (under typical conditions) collected at WCS, the mapping between the two power ratios was seen to be approximately

$$C_b/N_o = 45\text{dB} \approx \text{SNR} = 2\text{dB}. \quad (2.4)$$

In this thesis, we shall use C_b/N_o to describe results of actual signals and SNR when describing simulation results. To avoid confusion when comparing the two, we shall map SNR to C_b/N_o .

Since we intend DAMA to be a SS BPSK system, we must not only describe data rates, but also chip rates and therefore PGs. Due to the nature of the DAMA service, it has been proposed that $R_b = 1$ kbps [3]. The PG, as defined in (2.1), will be another parameter that is of primary concern since it determines the BW of the DAMA carrier and impacts the sampling rate. DAMA carrier estimation

accuracy will be shown to depend on PG to a large extent and therefore the chip rate will be a matter of investigation.

Sampling rate is another important parameter. It was initially proposed that the PG would be set to $PG = 100$ [3]. With the data rate set as above, this implies a chip rate, $R_c = 100$ kbps. This will exhibit a mainlobe width of 200 kHz by (2.3). From Figure 2.2 we can observe that we will need to account for another 100 kHz due to Doppler shift. The total possible BW of the DAMA carrier is then 300 kHz. We recall that to avoid aliasing while sampling the DAMA carrier, the DAMA signal must be bandlimited and be sampled at

$$f_s \geq 600\text{kHz}. \quad (2.5)$$

A commonly available, inexpensive 800 kHz A/D was found that matched the requirements and we thus chose $f_s = 800$ kHz. This has implementation ramifications that will be discussed below.

2.5 Carrier Estimation Problem

The problem with demodulating the DAMA carrier is the same as that for the MA carrier (though for the MA case the estimate is derived from the state vector of the satellite): the ground station is not capable of demodulating a signal if the error of the estimate is greater than ± 3 kHz from the actual. As NMSU's proposal will not keep track of state vector information for each user spacecraft, the carrier must be estimated reliably and efficiently and then passed to the ground station

receiver. NMSU's DAMA proposal hinges upon accurate carrier estimation. We have developed an algorithm and hardware that performs this task and we shall describe the algorithm as well as the parameters for operation that will provide accurate carrier estimation. The hardware will provide a locking tone to the ground station receiver.

2.6 Proposed Solution

The proposed solution relies on classical spectral estimation theory with some modifications to improve performance. We begin by assuming that external analog hardware required to bandlimit and frequency shift to baseband the TDRSS and DAMA signals is available. Contained in the 400 kHz band (assuming $f_s = 800$ kHz) will be a portion of the TDRSS signal along with the entire DAMA signal.

We next employ an averaged DFT (implemented with an FFT) which, if magnitude squared values are computed, is also known as a periodogram. Since we assume AWGN with zero mean, the averaging has the effect of reducing estimation error variance of the carrier based on the DFT estimate of the DAMA spectrum. We express this as

$$X(k) = \frac{1}{P} \left\{ \sum_{p=1}^P \left| \sum_{n=0}^{N-1} x[n + pN] e^{-\frac{j2\pi kn}{N}} \right|^2 \right\} \quad (2.6)$$

where P is the number of blocks in the average, N is the number of points in the block, n is the sample index, and k is the frequency index [9]. To obtain a frequency resolution that will enable us to estimate within the accuracy of the

GSR, we choose

$$N = \frac{f_s}{\Delta f} = \frac{800\text{kHz}}{3\text{kHz}} \approx 267 \quad (2.7)$$

but this would not allow us to use the *radix-2* based FFT. We alternatively round to $N = 512$ for use with the FFT which yields

$$\Delta f = \frac{f_s}{N} = \frac{800\text{kHz}}{512} = 1562.5\text{Hz} \quad (2.8)$$

This increases our physical resolution beyond what is actually required.

From the result of the periodogram, we are left with an estimate of the spectrum of the received signal. We estimate the carrier frequency by choosing the maxima of the periodogram. We assume that we are operating the DAMA service such that a peak will be observed in the average. We can improve the *spectrum-to-noise ratio* by utilizing a method from communication theory. It is known that the optimal solution for a receiver corrupted by AWGN is obtained by implementing a matched filter [7]. We will employ the frequency domain equivalent, spectral matched filter (SMF), which like its time domain analog is optimal and will maximize the SNR. In the time domain, a matched filter can be described by its impulse response

$$h(t) = s(T - t) \quad (2.9)$$

with

$$0 \leq t < T$$

where $s(t)$ is the time reversed equivalent of the received signal. The matched filter is then convolved with the received signal yielding

$$y(t) = \int_0^t s(\tau)s(T - t + \tau)d\tau. \quad (2.10)$$

The same approach may be applied in the frequency domain where we have a SMF that is matched to the expected, frequency-reversed Power Spectral Density (PSD) of the DAMA carrier. Through a discrete convolution between the SMF and estimated DAMA spectrum, we may arrive at a desired optimal solution that maximizes the spectrum-to-noise ratio. The SMF is described as

$$H(k) = X(N - k) \quad , 0 \leq k < N \quad (2.11)$$

where $X(N - k)$ is the frequency-reversed equivalent of the PSD of the DAMA carrier. The SMF is then convolved with the spectrum of the received signal as

$$\mathbf{X}_{smf} = \mathbf{X} * \mathbf{H} \quad (2.12)$$

where $*$ indicates the convolution operator. The results of applying the SMF to the process will be described in detail in section three. We then form our carrier estimate with

$$\hat{f} = \frac{\arg \max(X_{smf})}{N} f_s \quad (2.13)$$

where \hat{f} is the estimated frequency of the carrier, N is the number of points of the FFT and f_s is the sampling frequency. The discrete convolution in (2.12) smears

the number of points by

$$L_{X_{smf}} = L_H + M \quad (2.14)$$

where $L_{X_{smf}}$ is the translated length of X_{smf} , L_H is the length of the SMF ($N = 512$), and M is the length of the periodogram ($N = 512$). We must subtract the maxima of the SMF, purposely located at $N/2$ - or 256, from the index obtained by $\arg \max(X_{smf})$. The SMF is generated in practice by generating a simulated DAMA carrier (without noise) with random data and averaging 1000 periodograms based on the simulated signal. The simulated signal is set with the PG that we wish to test. A flowchart that describes the algorithm is shown in Figure 2.6.

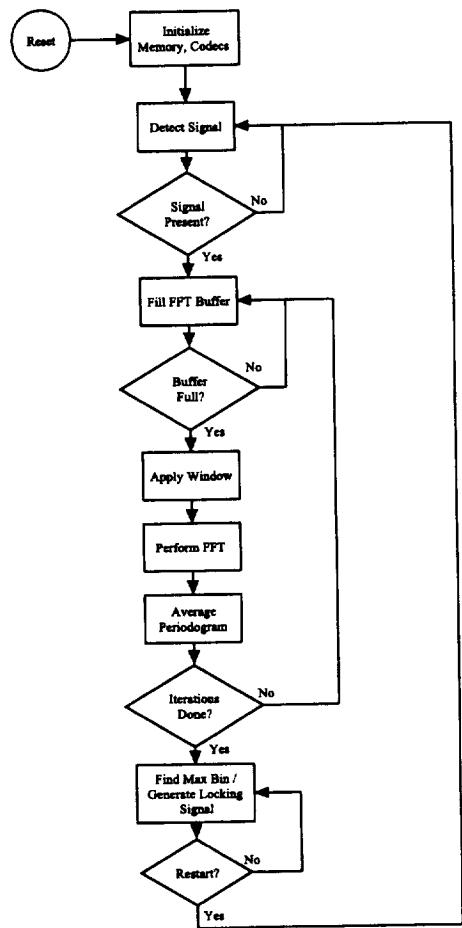


Figure 2.6: Flowchart for Carrier Estimation Algorithm

3 Theory and Simulation

3.1 Signal Description

We have previously stated that the DAMA carrier will use SS BPSK and we begin with a description of the communication scheme. BPSK is a communication scheme where data is encoded into the phase of a sinusoid (carrier). The general equation for M -ary PSK is shown in (3.1)

$$s(t) = A \cos(\omega_c t + \psi_m(t)) \quad (3.1)$$

where A is the amplitude of the carrier, ω_c is the carrier frequency and $\psi_m(t)$ is the phase component. In general we may add more power to the sinusoid by increasing A but we assume $A = 1$ for the purposes of this thesis. $\psi_m(t)$ may take on m distinct phases based upon the mapping of data to phase. For the binary case we use two distinct phases. Though we could choose any two distinct phases, in order to minimize error we typically choose the antipodal signaling as seen in Figure 3.1 as it provides the optimal decision boundary [8]. With this configuration, the phase term $\psi(t)$ may take on values corresponding to $\psi(t) \in \{0, \pi\}$. This is equivalent to modulating the sign of the sinusoid such that we may revise (3.1) to

$$s(t) = A d(t) \cos(\omega_c t) \quad (3.2)$$

where $d(t) \in \{-1, 1\}$ and is dependent on the mapping of data. As explained in Section 2.3, the data is DS spread by a PN code. For purposes of the analysis that

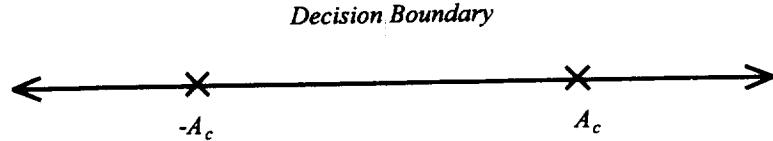


Figure 3.1: Signal Space Diagram for BPSK

follows, we may look at SS BPSK as just BPSK at a higher data rate, namely the chip rate. We make this assumption since we are interested only in searching for a spectral peak of magnitude-squared data and not in demodulating the signal.

3.2 Estimation Accuracy Theory

The following arguments follow closely the work of Boaz Porat in his book, A Course in Digital Signal Processing. We have only extended his arguments to frequency estimation of a BPSK signal [4].

To begin, we make the following assumptions: $s(n)$ is a BPSK signal as described in (3.2), $S(e^{j\omega n})$ is the DTFT of $s(n)$, $w(n)$ is the rectangular windowing function, and $v(n)$ is AWGN of zero mean. The received signal would then be

$$x(n) = s(n)w(n) + v(n)w(n)$$

taking the Fourier Transform of this yields

$$X(e^{j\omega n}) = \sum_{n=0}^{N-1} s(n)w(n)e^{-j\omega n} + \sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega n}$$

$$\begin{aligned}
&= \sum_{n=0}^{N-1} s(n)e^{-j\omega n} + \sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega n} \\
&= S(e^{j\omega n}) + \sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega n}
\end{aligned}$$

The PSD of $s(n)$ is given by [8]

$$P_s(e^{j\omega n}) = E[S^2(e^{j\omega n})] = \frac{A^2}{4R} \left[\text{sinc}\left(\frac{(\omega - \omega_c)\text{osf}}{2\pi}\right) \right]^2 \quad (3.3)$$

where we have transformed (3.3) from continuous time to discrete time, R is the data rate in bits/sec, and osf is the oversample factor in samples/bit. The maxima of the sinc function as well as the maxima of the BPSK periodogram will occur at $\omega = \omega_c$ hence

$$X(e^{j\omega n})|_{\omega=\omega_c} = S(e^{j\omega_c}) + \sum_{n=0}^{N-1} v(n)w(n)e^{j\omega_c n} \quad (3.4)$$

and taking the magnitude-squared of both sides,

$$\begin{aligned}
|X(\omega_c)|^2 &= \left(S(e^{j\omega_c}) + \sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega_c n} \right) \left(S(e^{j\omega_c}) + \sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega_c n} \right)^* \\
&= \left(S(e^{j\omega_c}) + \sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega_c n} \right) \left(S^*(e^{j\omega_c}) + \sum_{n=0}^{N-1} v(n)w(n)e^{j\omega_c n} \right) \\
&= |S(e^{j\omega_c})|^2 \\
&\quad + 2\Re \left[S(e^{j\omega_c}) \sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega_c n} \right] \\
&\quad + \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} w(n)v(n)w(m)v(m)e^{-j\omega_c(n-m)}
\end{aligned}$$

Taking the expectation of the magnitude-squared value yields,

$$E[|X(\omega_c)|^2] = E[|S(e^{j\omega_c})|^2]$$

$$\begin{aligned}
& + E \left[2\Re \left[S(e^{j\omega_c}) \sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega_c n} \right] \right] \\
& + E \left[\sum_{n=0}^{N-1} \sum_{m=0}^{N-1} w(n)v(n)w(m)v(m)e^{-j\omega_c(n-m)} \right] \\
& = P_s(e^{j\omega n}) \\
& + 2\Re \left[S(e^{j\omega_c}) \sum_{n=0}^{N-1} E[v(n)]w(n)e^{-j\omega_c n} \right] \\
& + \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} w(n)v(m)E[v(n)v(m)]e^{-j\omega_c(n-m)}
\end{aligned}$$

The first term is by definition as provided in (3.3) and is evaluated as

$$P_s(e^{j\omega})|_{\omega=\omega_c} = \frac{A^2}{4R}$$

The middle term evaluates to 0 since we have assumed $E[v(n)] = 0$. We further assume $E[v(n)v(m)] = \gamma_\nu \delta(n - m)$, then

$$\begin{aligned}
E[|X(\omega_c)|^2] & = \frac{A^2}{4R} + \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} w(n)w(m)\gamma_\nu \delta(n - m)e^{-j\omega_c(n-m)} \\
& = \frac{A^2}{4R} + \gamma_\nu \sum_{n=0}^{N-1} w^2(n) \\
& = \frac{A^2}{4R} + \gamma_\nu N
\end{aligned}$$

We next define an output SNR using (3.7) as the ratio between

$$SNR_O = \frac{\frac{A^2}{4R}}{\gamma_\nu N} \quad (3.5)$$

We define the input SNR as

$$SNR_I = \frac{\frac{A^2}{2}}{\gamma_\nu} \quad (3.6)$$

where $A^2/2$ is the average power of a BPSK signal [8]. We delineate between output and input SNR to account for the application of the window function.

Finally we define a window processing gain as

$$W_g = \frac{SNR_O}{SNR_I \frac{1}{2}N} \quad (3.7)$$

$$= \frac{2}{N} \left[\frac{A^2}{4R\gamma_\nu N} \frac{\gamma_\nu}{\frac{A^2}{2}} \right] \quad (3.8)$$

$$= \frac{1}{RN^2} \quad (3.9)$$

We next employ a rule of thumb which states

$$\frac{NA^2W_g}{\gamma_\nu} \geq 100 \quad (3.10)$$

which is given in Porat as a requirement for the reliable detection and frequency estimation of a real sinusoid in noise [4]. We modify (3.10) for the BPSK case in terms of (3.5), (3.6), and (3.9) which yields

$$\frac{A^2}{RN\gamma_\nu} \geq 100 \quad (3.11)$$

In the development provided by Porat, the mean square error is then given as

$$E[\hat{f}_c - f_c]^2 \approx \frac{24N_o J_w}{(2\pi)^2 A^2 D^3} \quad (3.12)$$

where N_o is the power density of the noise, J_w is a window parameter [4], D is the measurement interval ($D = NT_s$, where T_s is the sampling period), and A is the amplitude of the sinusoid. We further make the assumption that $E[\hat{f}_c - f_c] = 0$.

With this assumption and provided that (3.10) is true, the approximation in (3.12) is valid [4]. The approximation given in (3.12) can now be modified to yield

$$E[\hat{f}_c - f_c]^2 \approx \frac{24N_o J_w}{(2\pi)^2 \frac{A^2}{4R} D^3} \quad (3.13)$$

where we have applied (3.3) evaluated at $\omega = \omega_c$. The result of (3.13) is an approximation of the mean square error (MSE) of the BPSK carrier frequency estimate. We note that like the sinusoid in noise case, the approximation in (3.13) is only valid when we ensure that (3.11) is true.

Equation (3.13) provides an approximation for the expected value of the MSE frequency estimate. We can use it to provide an approximation of the error in estimated frequency \hat{f}_c from a true frequency f_c from using a DFT approach to estimation. It is recognized that with the many assumptions and the inclusion of a rule of thumb that (3.13) can only give a rough approximation to the actual MSE of the frequency estimation [4].

3.3 Comparison of Theory to Simulation

Having obtained the MSE, we now show how (3.13) compares to simulation data. In the simulation we modeled a BPSK signal with a carrier frequency, f_c , of 178 kHz (typical of DAMA) and the sampling frequency, f_s , set to 800 kHz (as in DAMA). The simulation performed an $N = 512$ point DFT on the BPSK signal and measured the MSE of the estimated frequency, \hat{f}_c , to the actual frequency f_c . The results of the simulation are shown in Figure 3.2 where we have shown the

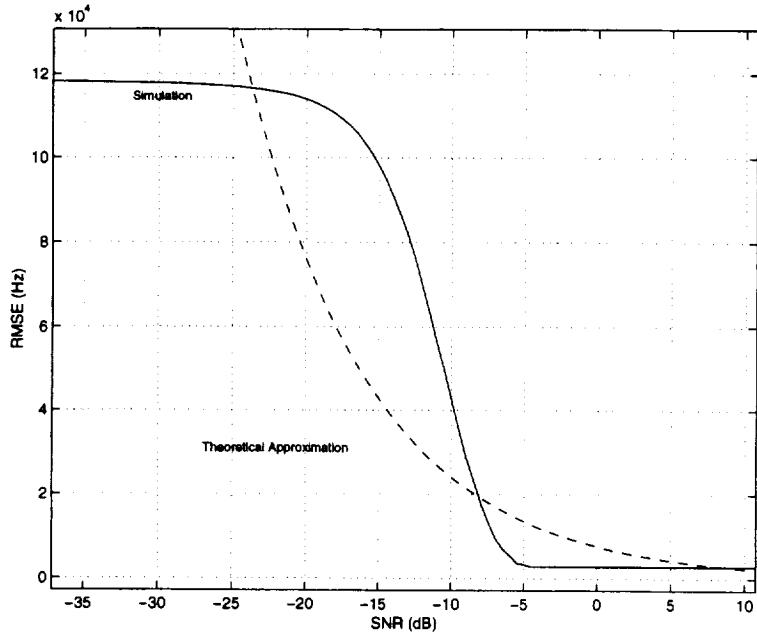


Figure 3.2: Comparison of Theoretical Curve Against Simulation

simulation results versus the approximation developed in (3.13). The root mean square error (RMSE) is simply the square root of (3.13) and it is the expected value in Hz that \hat{f}_c is from f_c as a function of SNR. This relates directly to the accuracy of estimating f_c with \hat{f}_c to within the GSR tolerance and provides a means by which we can approximately determine the accuracy for a given SNR.

We can conclude from the simulation that at high SNR's (< 5 dB), the theoretical curve and the simulation curve match, while at low SNR's the theoretical curve becomes invalid (due to assumptions made during calculation) [4]. We further conclude that at high SNR's (3.13) will provide a good estimate of the mean of the estimation accuracy and allow us to predict performance. At low

SNR's the simulation curve approaches a condition where the noise is dominating the spectrum and the maximum peak is uniformly distributed over the 400 kHz band, yielding near random estimates. The theoretical curve is asymptotic to zero RMSE whereas the simulation curve will actually converge only to the difference between the true frequency and the selected DFT frequency—the difference is unlikely to be zero. This illustrates an important point regarding accuracy of carrier estimation: since we quantize to the DFT frequency points, there will most likely be an error irrespective of what SNR we are operating at. The maximum amount of this error assuming large SNR is given by

$$\max(f_c - \hat{f}_c) < \frac{f_s}{N} \quad (3.14)$$

This error can only be reduced by increasing N , the number of points of the DFT.

3.4 Simulation Model Description

Though we have developed a theoretical analysis culminating in (3.13), the approximation is not tight enough to provide proof of concept. We turn to a Matlab simulation model to provide a core proof of concept. We shall use this simulation model to explore the operational parameters of the algorithm. The simulation models the DAMA carrier against other MA users and AWGN. The simulation builds up a SS BPSK digital waveform based on the PG that we wish to test and then adds appropriate AWGN. It then estimates the doppler-shifted carrier frequency based upon the algorithm developed in section two. A record is

kept of carrier frequency estimations and those that fall within the tolerance of the ground station at WSC are counted as an accurate estimation. Likewise any estimation that is outside of the ground station tolerance is considered inaccurate. By performing the simulation 10,000 times for each SNR and desired PG, we obtain a plot that describes estimation accuracy.

The simulation was originally written without the SMF and then rewritten to include the SMF process. The SMF improves estimation accuracy and allows for higher PG's. The Matlab code for the simulation is given in Appendix A.

3.5 Simulation Results

We now provide results of simulation both in the SMF and non-SMF cases. The simulation results for the non-SMF are shown in Figure 3.3 and illustrate estimation accuracy as a ratio of accurate estimations to total estimations versus SNR. In Figure 3.3, we plot three curves for PG = 10, 20, and 100. The curves are generated through 10000 estimates per SNR and we perform 8 DFT blocks of $N = 512$ points. At an SNR = 2 dB, the results of Figure 3.3 demonstrate that estimation accuracy would be approximately 14%, 60%, and 88% for PG = 10, 20, and 100 respectively. NASA has placed the requirement that any implementation

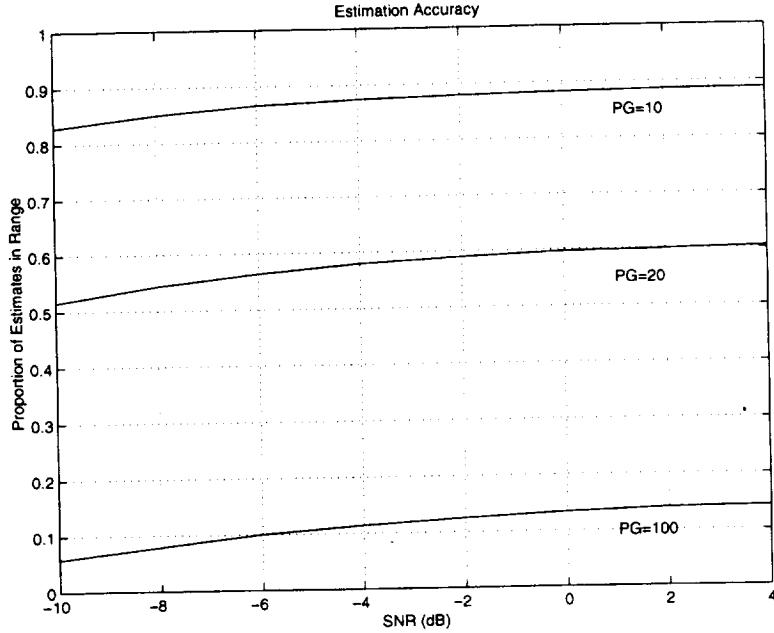


Figure 3.3: Family of Curves for Non-SMF Case

should have as high a PG as possible. From Figure 3.3 it is clear that only the PG = 10 case is practical.

Carrier frequency estimation may also be observed through the use of histograms which also relate information regarding the variance of the estimations. In Figure 3.4, Figure 3.5, and Figure 3.6 we show histograms for PG = 10, 20, and 100 respectively. Each is shown for SNR = 2. In each figure the actual carrier frequency is denoted by the center dashed line while the GSR tolerance (± 3 kHz) is shown by the outer dashed lines.

The results with the SMF enhancement, as described in section two, are much better (Figure 3.7). The results shown in Figure 3.7 are generated as in the

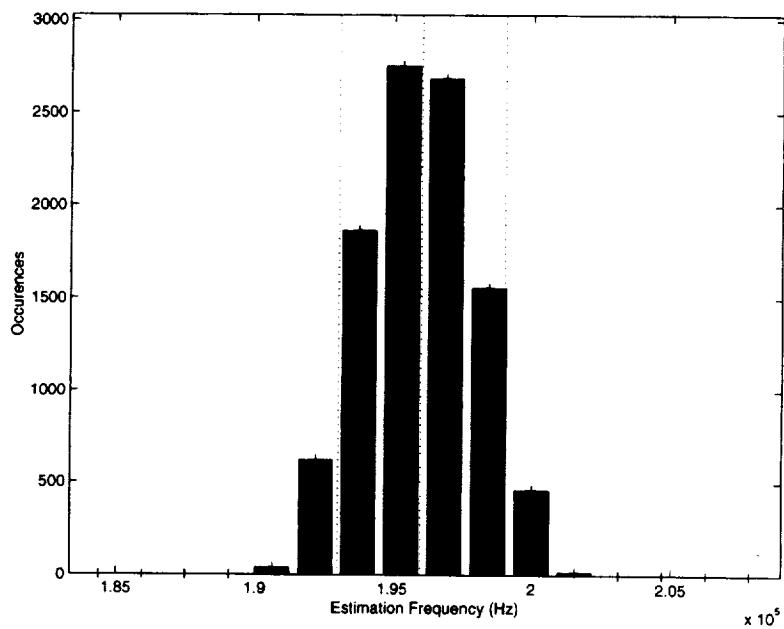


Figure 3.4: Non-SMF Frequency Estimation for $PG = 10$, $SNR = 2$ dB

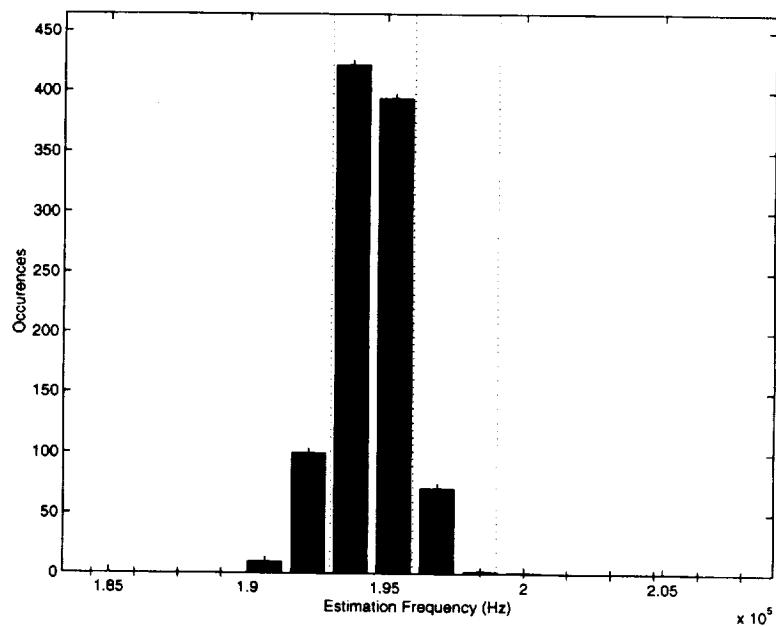


Figure 3.5: Non-SMF Frequency Estimation for $PG = 20$, $SNR = 2$ dB

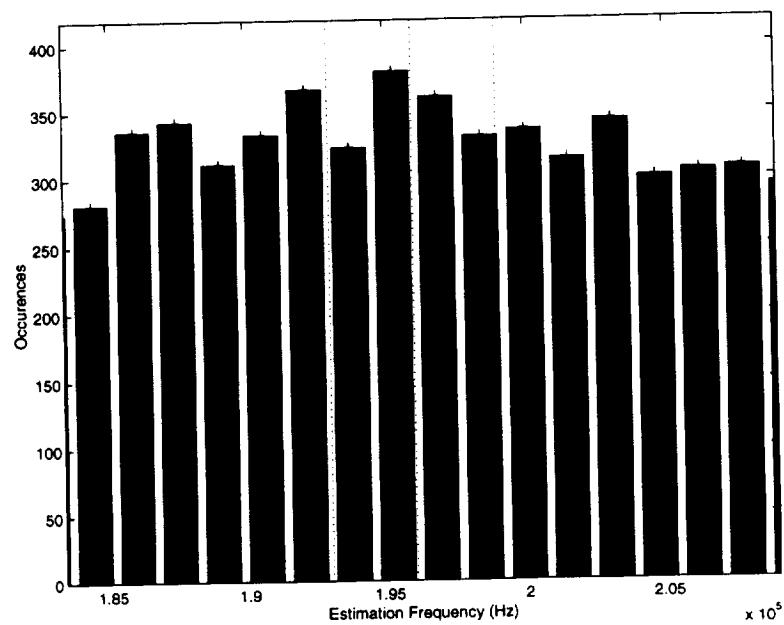


Figure 3.6: Non-SMF Frequency Estimation for $PG = 100$, $SNR = 2$ dB

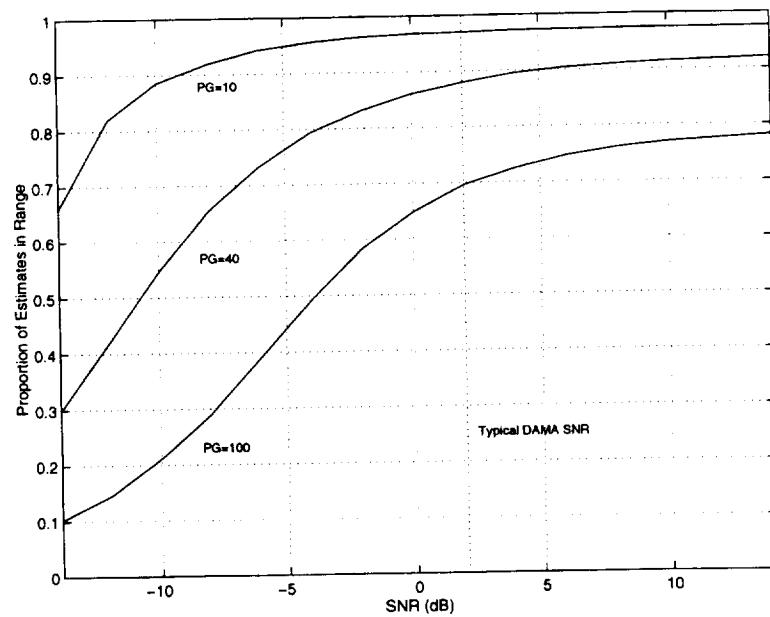


Figure 3.7: Family of Curves for SMF Case

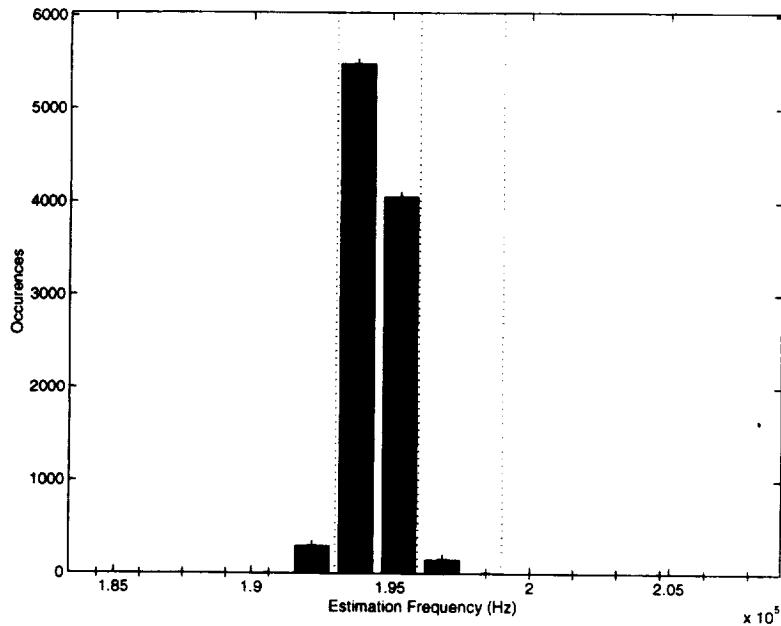


Figure 3.8: SMF Frequency Estimation for $PG = 10$, $SNR = 2$ dB

non-SMF case with the exception that we have now included SMF processing. In Figure 3.7, we show estimation accuracy curves for $PG = 10, 40$, and 100 . Here we see that estimation accuracy of the $PG = 100$ case has improved seven-fold. The case of $PG = 40$, a likely DAMA operating parameter, demonstrates accuracy at nearly 90% in a typical operating SNR region. As in the non-SMF case, we observe the histograms of Figure 3.8, Figure 3.9, and Figure 3.10 which are shown for $PG = 10, 40$, and 100 respectively. Each figure is shown at $SNR = 2$. From these figures it is clear that there is less variance and therefore more estimations within the GSR tolerance.

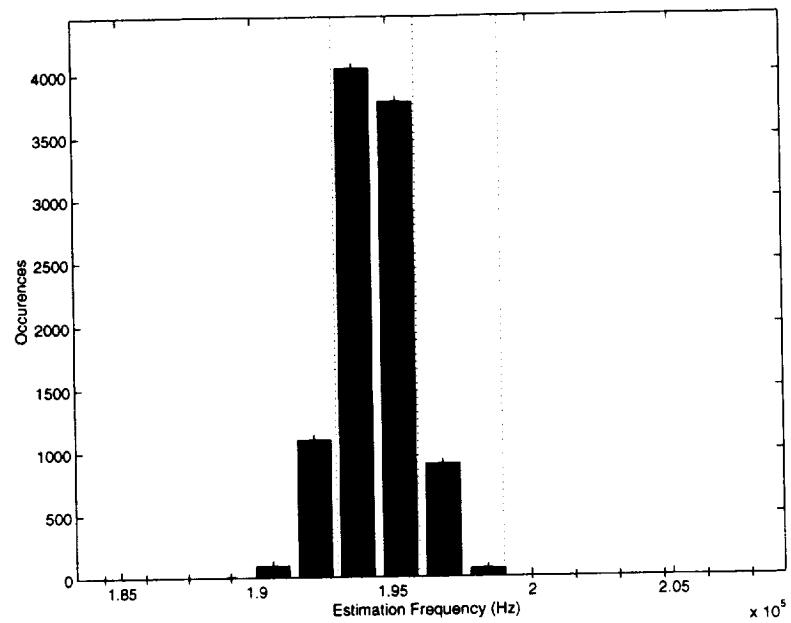


Figure 3.9: SMF Frequency Estimation for $PG = 40$, $SNR = 2$ dB

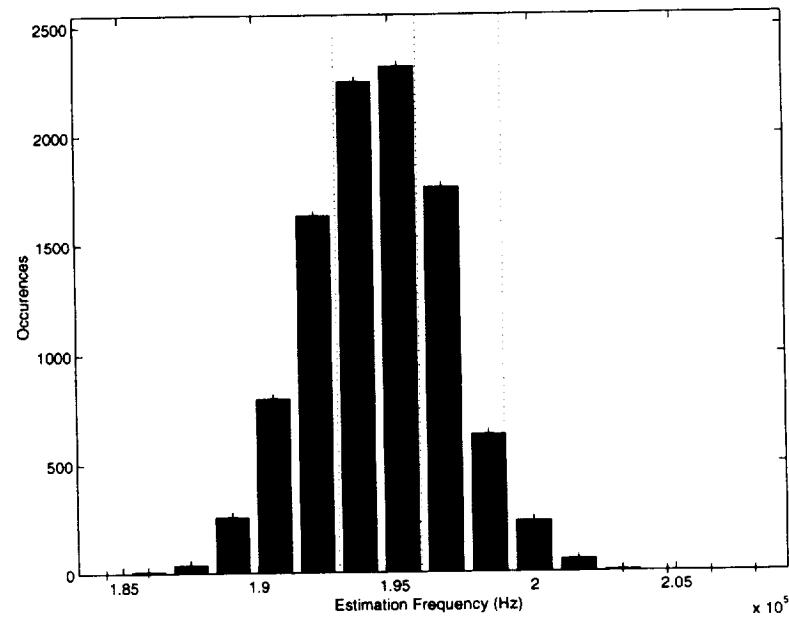


Figure 3.10: SMF Frequency Estimation for $PG = 100$, $SNR = 2$ dB

3.6 Theoretical/Simulation Data Summary

We have demonstrated results for a model that attempts to simulate DAMA carrier estimation in the presence of the TDRSS spectrum which is modeled with AWGN. We provide results indicating that accurate carrier estimation is possible with the algorithm developed in section two. Additionally, we have determined some operating points for the PG parameter. We have shown that we can accurately estimate the DAMA carrier 90% of the time with a $PG = 40$ at the typical SNR range. The simulation has provided a demonstration and we offer a loose theoretical approximation (3.13) to the estimation accuracy. In section four, we shall use data collected at NASA's White Sands Complex (WSC) to further validate the simulation results.

Appendix A includes all of the developed code for the simulation model including instructions for its use. Additional code is provided to perform data visualization.

4 WSC Data Collection Experiment

4.1 Motivation of WSC Data Collection Experiment

An experiment was devised to perform validation of the simulation model by collecting actual signals transmitted through TDRSS and collected at WSC. We subsequently processed them offline using the estimation algorithm. The fundamental idea of this experiment is that we can observe the performance of the algorithm with actual DAMA signals. If the algorithm using actual signal data performs similar to the simulation under various PGs and at various SNRs, we may state with a degree of certainty that the simulation model is indeed accurate enough to predict carrier estimation. This also allows for the prediction of the performance of the algorithm if any of the operating parameters need to be changed. A side benefit is that we may also perform *in-house* testing using simulated signals without the expense of interrupting critical TDRSS operations for testing.

This section discusses the actual experiment performed at WSC, including details regarding setup, the processing of the data, and the conclusions drawn. We verify the simulation model's results with actual TDRSS data. Additionally, the experiment points out the realistic operating boundaries which are an important part of the DAMA design.

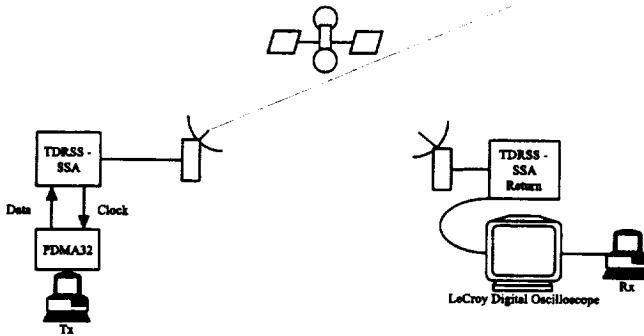


Figure 4.1: Experiment Setup

4.2 Data Collection Setup

Two sets of equipment were used to collect test signals that were sent from the ground station and relayed off of a TDRS back to the ground station. The setup was as shown in Figure 4.1. The Transmit (Tx) equipment consisted of a computer equipped with a high speed PDMA32 Data Transfer Card (PDMA32). The purpose of the Tx equipment was to send a data vector to ground station equipment where it was BPSK modulated and transmitted to a TDRS. The data vector sent consisted of underlying data bits of $\{\pm 1\}$ with $R_b = 1$ kbps which was then spread by a spreading vector at a rate of $R_c = PG \cdot R_b$ chips/s. Several of these data vectors, consisting of raw binary data and DS spread with a PN code (see section two), were generated in advance of the experiment and then used as a data source. The source code (wsands.m, wsands2.m, wsands3.m, and wsands4.m) is given in Appendix A.

The Receive (Rx) equipment consisted mainly of a LeCroy Digital Storage Oscilloscope (DSO) to capture IF signals and another computer to store captured signals. As the TDRSS channel is bandlimited to approximately 40 MHz, the DSO was set to sample at a rate of 100 MHz. This was the closest value to the Nyquist rate that the DSO was capable of sampling at. The DSO was capable of storing the captured data waveforms with either 50,000 samples or 100,000 samples depending on the storage medium that was used and both sizes were collected.

With the hardware setup described, we now turn our attention to the test set. The test set was established to test a variety of key parameters and determine the operational bounds of each. The key parameters are

- Processing Gain
- DAMA Carrier power-to-noise ratio (C_b/N_o)
- Placement of IF Carrier Frequency against TDRSS spectrum

The PG, defined in (2.1), is the most critical of the three. Initial work with the PG indicated a PG of 100 could be used [3]. Subsequent simulations indicated that at this PG value the estimation accuracy was not reliable and lower values were investigated (see section three). It was included in the test set for completeness. PG's of 10, 20, 40, and 100 *chips per bit* were the focus of the test set.

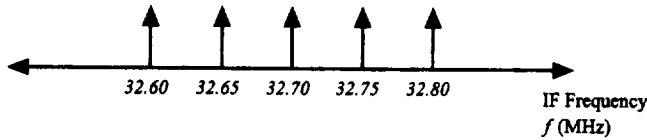


Figure 4.2: Placement of Test Frequencies Near TDRSS Null

Three different carrier power-to-noise ratios were investigated. These ratios were 40, 45, 50 dB. These values were chosen as typical values based upon the experience of WSC staff [10]. It should be noted again that these values are described as a ratio of the carrier power to the noise floor in dB and not carrier power to MA spectrum as described in section three.

The last test parameter is that of the carrier placement. The carrier frequencies were chosen such that a range of frequencies near the TDRSS null could be examined. Recall from section two that we intend to place the DAMA carrier at 2.29 GHz which at IF is 2.9 MHz above the TDRSS center frequency. The range of test frequencies are observed in Figure 4.2. By examining a range of IF frequencies, we are allowed to effectively simulate a Doppler shift as well as examine the effects of the rolloff of the TDRSS channel. Comparing this to the TDRSS spectrum shown in Figure 2.2, we see that the upper range will be affected by the TDRSS rolloff. The full test set is shown in Table 1. The full test set had to be reduced for logistical reasons to lessen impact on TDRSS operations, however, the reduced test set provides enough insight for simulation verification.

Table 1: Test Sets Captured

		C_b/N_o		
PG	F_c MHz	40 dB	45 dB	50 dB
10	32.60		*	
	32.65	*	*	*
	32.70	*	*	
	32.80		*	
20	32.60			
	32.65	*	*	*
	32.70		*	
	32.80			
100	32.60			
	32.65		*	
	32.70			
	32.80			

* indicates data collected

4.3 WSC Collected Data Processing

We noted above that the captured waveforms were sampled at 100 MHz and stored as either 50,000 or 100,000 length vectors. In section two it was also noted that the bandwidth of the DAMA carrier plus maximum possible Doppler shift yielded a 300 kHz frequency search space and that the FFT resolution will be 1562.5 Hz. Downsampling the captured waveforms to $f_s = 800$ kHz was impractical due to the problems of designing a narrow band decimation filter sharp enough for the rate conversion. Instead we seek a solution that will allow us to perform a frequency search with a nearly equivalent FFT resolution. We see then that

$$\Delta f_{experiment} = \frac{f_s}{N} = \frac{100MHz}{65536} = 1525.8Hz \quad (4.1)$$

provides a nearly equivalent FFT resolution to (2.8). In order to provide the

resolution in (4.1) and maximize the use of the short data sets, the vectors were either *overlapped* in the 100,000 sample case or *zero-padded* in the 50,000 length case. Without zero-padding the 50,000 length vector to 65536, the DFT does not meet the required resolution. The net result of this operation is a collection of 65536 length vectors with which to work with. These vectors were then used in a periodogram of eight data blocks chosen at random and processed with the simulation code that was used with a synthetic waveform (see Appendix A). This process was iterated 500 times, due to maximum data support, to form simulation bounds using WSC data.

4.4 WSC Data Results Compared to Simulation

The results of the processed data in comparison to simulation results are shown in Figure 4.3 below. The plot shown is for $PG = 10$ with the simulation represented by the curve and the individual data points being the collected data across the various frequencies indicated. The plot indicates successful carrier estimation to within ± 3 kHz as the ratio of accurate estimates to total estimates versus C_b/N_o . We can make several observations from the plot.

The first observation we can make is in regards to carrier power C_b/N_o . We observe that at a $C_b/N_o = 40$ dB, the estimation accuracy is much less than the simulation exhibits indicating a practical limit to C_b/N_o . As C_b/N_o increases to 45 and 50 C_b/N_o , estimates performed with actual WSC signal data match

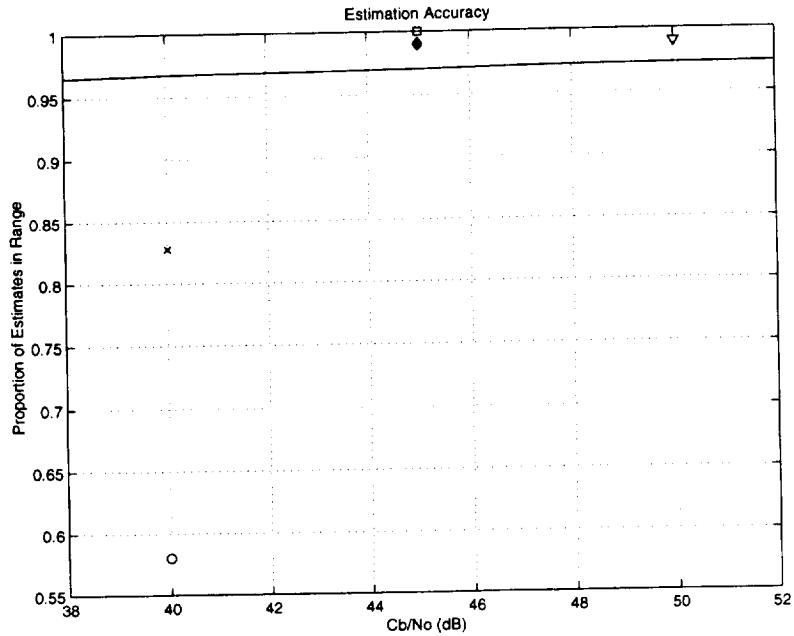


Figure 4.3: Comparison of Simulated Results vs. WSC Captured Data

those using synthetic data. This result indicates that we can estimate Doppler accurately provided that the carrier power is high enough for the range of carrier frequencies specified in Figure 4.2. This is an intuitive result as well since we would fully expect that a carrier with higher power will exhibit a more defined peak against the TDRSS spectrum. We also observe that the nominal carrier power provided by WSC will be sufficient as an operational bound. Previous work indicated that the DAMA carrier power and the MA carrier power can be relatively the same [3]. For the algorithm developed however we must have enough power so that the DAMA carrier exhibits a peak against the MA spectrum. Based

on collected data for the $C_b/N_o = 40$ case, the DAMA carrier is lost in the MA spectrum and in general cannot meet reliability requirements.

4.5 Conclusions of WSC Data Collection

We have shown some important results with the WSC Data Collection analysis. The algorithm using actual TDRSS signals performs nearly the same as when using synthesized signals (see Figure 4.4). Therefore the synthesis of modeled signals is accurate to actual TDRSS signals. We rely heavily on the simulation to provide proof of concept and so the justification of the simulation is vital: the simulation is used to narrow in on the operating parameters that we should like to run at. Though we only have significant data for the $PG = 10$ case, we extend the results described above to higher PG's with the assumption that the effects of spreading the signal even further in the frequency domain will cause negligible disparities between the simulation and actual operation. We can use simulated waveforms to implement additional, and important, real-time hardware tests.

There remains one outstanding issue with regards to accurate carrier estimation. We have seen previously that the TDRSS channel begins to rolloff sharply near the location that we would like to place the DAMA carrier. Though the data did not support the investigation—at least in the spectral matched filter case—it is nevertheless true that if the DAMA carrier is nominally placed too close to the

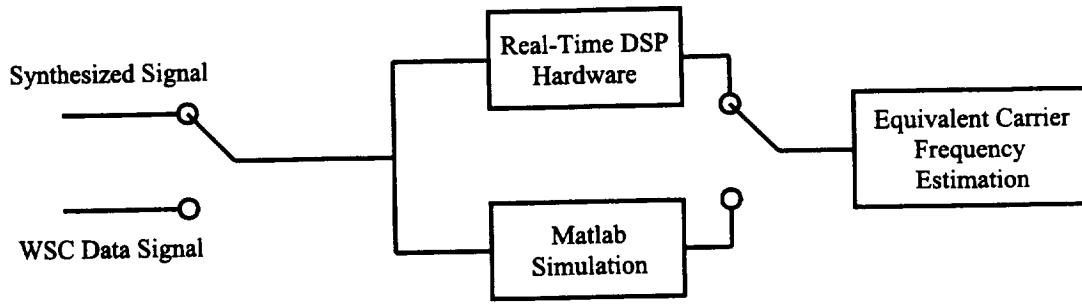


Figure 4.4: Carrier Estimation Summary

null, then the rolloff will adversely affect accuracy. We must also avoid placing the DAMA carrier in the MA passband.

Overall the successful comparison of captured data to simulation data indicates that provided the DAMA carrier has sufficient power, we can explore additional operation bounds with a high degree of confidence.

5 Carrier Estimation Hardware and Software

5.1 Motorola DSP56303EVM Description

The *DSP56303EVM*(EVM) is the core hardware component of the carrier estimation hardware developed. It executes the DSP specific real-time assembly language version of the carrier estimation algorithm (see Appendix B). The EVM is an evaluation module which is designed to be used in prototyping applications. As such, it offers a variety of interface options and configuration settings that make it adaptable to many different types of development projects. The board contains a DSP56303 24-bit digital signal processor that executes the assembly code routines. It also features a DSP56002 specifically for use in I/O functions through a JTAG/OnCE port. JTAG is a protocol that was developed to allow hardware and software developers to observe and manipulate hardware for troubleshooting purposes. The JTAG port for the EVM is primarily used with a PC and debugger software that allows a developer to load code into the EVM, single step through sections of code, and observe the contents of memory and registers. The onboard DSP56002 is not available for coprocessing code, but rather is used to control the I/O functions to the host PC. The EVM also contains 32K x 24-bit Fast Static Ram (SRAM) built with three banks of 32k x 8 bit SRAMS with 15 ns access times. An additional 64k x 8 bit Flash Programmable Erasable Read-Only Memory (PEROM) is provided for stand alone operation. The EVM also contains

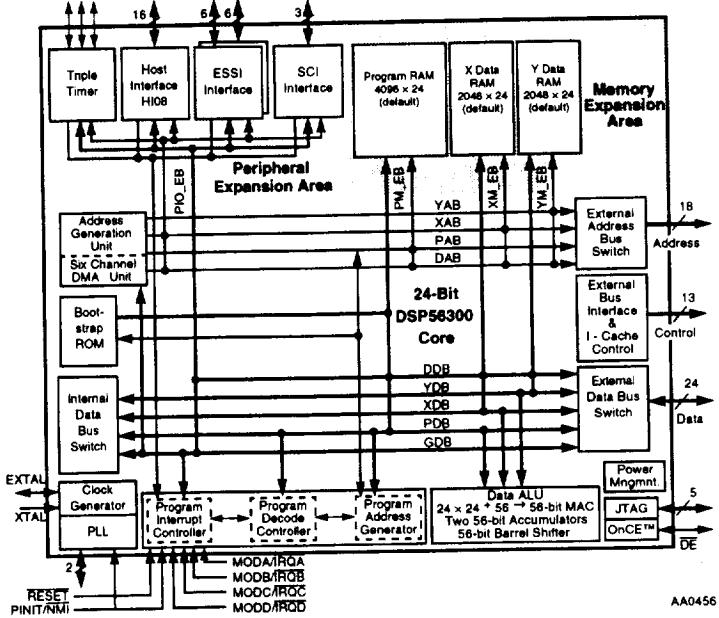


Figure 5.1: DSP56300 Core System Block Diagram

a 16 bit CODEC for sampling incoming analog waveforms and producing analog waveforms out from digital data [11].

The EVM is based upon the DSP56300 core which is shown in Figure 5.1. The EVM specifically uses the DSP56303 processor. The processor is capable of 80 MIPS with an 80 MHz clock. It provides for backwards compatibility with 56k core code so that code written for earlier processors should function equally well on the newer cores. Due to a seven level instruction pipeline architecture, the DSP56303 is capable of effectively an instruction every clock cycle. It is based on the *Harvard* architecture so that it works with a dual memory structure. This architecture is particularly suited for parallel move instructions. This allows the

programmer to access two different memory structures at a time. For more on the DSP56303 please see [12].

The DSP56303 not only provides the processing power required to perform carrier estimation efficiently, but it is highly configurable and can interface to a large number of peripheral products. Though higher level languages (C) may be used to program the processor, it is most efficient when programmed in assembly. The assembly language that the DSP56303 uses is specialized to perform DSP tasks and includes features for that purpose. The Real-Time version of the carrier estimation algorithm is developed completely in assembly (see Appendix B).

To create a program for the DSP56303, code is written in assembly language, and then assembled through the use of Motorola software tools to machine usable object code (.cld files). We may then use a software tool like Domain Technologies Debug-EVM to load the code into program memory of the DSP56303. With the Debugger software, we can utilize the JTAG port to single step through code and observe the results on calculations.

For the DAMA project, as was described in section two, we are required to sample at a higher rate than the EVM can perform with the on board CODEC. We will use the on board CODEC in the generation of the locking tone to the ground station receiver. To sample the DAMA spectrum, we need to interface a high speed A/D.

5.2 Burr Brown 800kHz A/D

Earlier we noted we must search over a 300 kHz BW (200 kHz for the DAMA mainlobe, assuming no more than a $PG = 100$, and with ± 50 kHz for maximum Doppler shift) which was revised from an earlier estimate of 364 kHz. The Burr Brown 12 bit 800 kHz ADS7810/19 (BB A/D) was chosen since it has a sampling frequency of 800 kHz which gives a Nyquist interval of 400 kHz and because it is available in a convenient evaluation package for easy interfacing to the DSP56303EVM. The 12-bit samples of the BB A/D have a dynamic range of -72 dB which is sufficient for the TDRSS system. It was necessary to build an interface board to allow samples collected with the BB A/D to be passed to the 303.

5.3 A/D Interface Board

The A/D Interface Board (ADIB) was developed as a Masters project by Tim Baggett [13]. Its purpose is to provide an interface to allow the EVM to communicate with the BB A/D. The board was developed to allow samples taken with the BB A/D to be passed into a peripheral (upper) memory location, which corresponds to the memory mapped I/O portion of memory. Though it is not a requirement, we may access the data samples with an efficient fast interrupt routine which is preferable. It is also possible with the interface board to use direct memory transfers (DMA) though this was not implemented in the Real-Time ver-

sion of the carrier estimation algorithm. The board allows for user configuration of the specific memory location the samples will be written in to. The ADIB was further complicated by voltage level discrepancies between the BB A/D and the 303. This was overcome through the use of *zero wait-state* level translators that adjust the output levels of the BB A/D to levels acceptable by the 303.

5.4 Additional Hardware

In an actual implementation of the DAMA carrier estimation hardware, there will be additional hardware required. Recall from the discussion of the DAMA project implementation, it was stated that WSC could provide the DAMA carrier at an IF of 32.65 MHz. The range of frequency values that a DAMA carrier can take, identified as 300 kHz, must be pre-filtered to bandlimit the signal containing the DAMA carrier from the MA waveform or aliasing will result. This is to be accomplished using an analog pre-filter set to a passband over the range of interest.

With the signal now appropriately pre-filtered, the signal must be *frequency shifted* to baseband utilizing additional commonly available analog hardware. With this pre-processing accomplished, the DAMA carrier can be sampled and the digital signal processed to detect and estimate the carrier frequency utilizing the algorithm described in section two. Upon the algorithm's determination of the DAMA carrier frequency, the CODEC on the EVM is utilized to provide a locking tone. By choosing a sample rate of 32 kHz for the synthesis of the locking tone, we

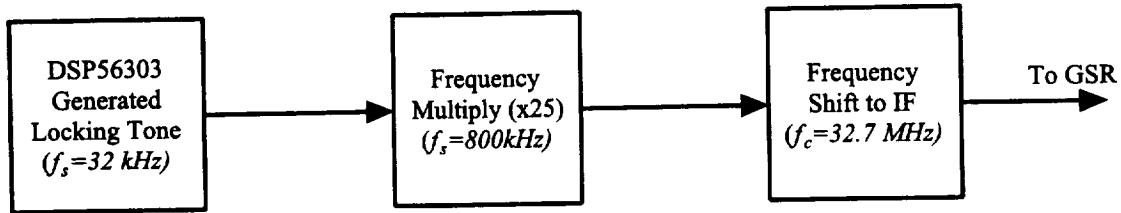


Figure 5.2: Locking Tone Generation

need only frequency multiply by 25 to get the appropriate frequency value (< 400 kHz). Afterward we frequency shift this tone back to IF for the GSR. This is illustrated in Figure 5.2. The analog hardware required to implement the frequency multiplication and shifting is commonly available and an essential component in implementation.

5.5 Software Description for Real-Time Carrier Estimation

The carrier estimation code is designed to work with the EVM and the ADIB to sample the incoming signal, process the signal, and then produce a locking tone to the ground station. The core of the program is based upon the code supplied with the EVM named **pass.asm**. The purpose of the **pass.asm** is to initialize the EVM (CODEC and processor), then transfer samples to/from the CODEC from/to the processor. All signal processing is performed between transfers. This code is important because in most cases of programming a Motorola DSP56xEVM, the algorithm to be implemented will use **pass.asm** as the starting point. The carrier estimation code is slightly modified in that while it uses **pass.asm** as the core, it must set the EVM to properly receive samples from the BB A/D

and instead of reading sample values from the on-board CODEC, it will instead read sample values from an upper memory location as described above. Upon completing the initialization of the EVM (CODEC and processor) to work with the BB A/D, the carrier estimation algorithm begins. We begin by looking at the flowchart in Figure 2.6 which describes the algorithm.

The initial step from the reset condition is to initialize memory and on-board CODEC and prepare the EVM for communication with the ADIB. The algorithm then begins by filling the sample buffer with values. Once it has achieved a full block of 512 samples, it applies a windowing function of the user's choice. Typically either Rectangular or Hamming window coefficients are used. Included in this window function is an iteration scaling factor of $1/P$, where P is the block number, which is nominally set to 0.125. This is included to scale for averaging. This has the added advantage of saving computation time since both windowing and average scaling are accomplished at once.

From the flowchart we now see that the actual FFT is performed. The FFT routine is supplied by Motorola (see Appendix B) and produces a normally ordered Fourier transform, as opposed to a bit-reversed, complex result from a normally ordered input. The result is stored such that the real component of the result is in X memory and the imaginary in Y memory. We compute the *magnitude-squared* of the FFT and repeat this step averaging magnitude-squared data $P = 8$ times to form the periodogram. If we assume that $x(n) \in \Re$ where $x(n)$

is the received, sampled signal, we need only search the first $N/2$ points due to Hermitian symmetry

$$X(k) = X^*(N - k), \quad 0 \leq k \leq N - 1. \quad (5.1)$$

This reduces the computational time and memory usage.

Continuing with the flowchart in Figure 2.6, once the periodogram is computed, we convolve the result with the SMF and begin a search for the maxima of the enhanced spectrum. The result of this search is an index corresponding to the dominant spectral component. From this index a frequency may be found according to the relation

$$\hat{f} = \frac{k - \frac{L}{2}}{N} f_s \quad (5.2)$$

where \hat{f} is the estimated frequency [14], k is the index found from the search, $L = N$ is the length of the SMF, N is the number of points of the FFT, and f_s is the sampling frequency. A sine wave table is then used to provide a locking tone at 1/25 frequency that the ground station expects. As described above, this locking tone will be frequency multiplied up IF. The DSP56303 assembly code is given in Appendix B.

6 Conclusions and Future Work

In this thesis we have described an algorithm to perform carrier estimation for the DAMA project. We have shown that the estimation accuracy depends primarily on the processing gain of the SS BPSK signal though some tertiary parameters also affect accuracy. We have developed a simulation model that provides proof of concept and provides DAMA operational parameters. Through the use of data collected at the WSC, we have verified the simulation model and established an operational point of $PG = 40$ with a corresponding accuracy of 90%.

In addition to the development above, we have developed Real-Time code based on the algorithm developed utilizing a Motorola DSP56303EVM. The entire project is implemented in hardware and functions equivalently to the simulation model.

Future work might involve the formulation of a tighter theoretical bound as well the development of the analog hardware required for implementation. This work could be applied in other applications relating to communications between user spacecraft and commercial telecommunication satellites. In this scheme, additional carrier pre-shifting would be required.

A Matlab Simulation Code

dama.m

```
001 %*****  
002 % DAMA dama  
003 %  
004 % Description: This function allows simulation of the estimation of a DAMA  
005 % user's Doppler-shifted carrier frequency. We assume a spread spectrum  
006 % system and the presence of other SS systems (modeled by AWGN).  
007 %  
008 % Programmer: Phillip De Leon  
009 %  
010 % Creation Date: December 31, 1995  
011 % Last Revision: Sept. 24, 1998  
012 %  
013 % Version History:  
014 % 1.0 - frequency estimate of sinusoid in noise  
015 % 2.0 - frequency estimate of DAMA carrier in noise (spread spectrum)  
016 % 3.0 - better memory management, more efficient  
017 % 4.0 - STFTs or average FFTs or MA filter smoothing of FFT bins  
018 % 5.0 - 800kHz sampling rate, 512-point FFT  
019 % 6.0 - frequency estimate now over 512 blocks (of a longer signal).  
020 % 7.0 - added frequency-domain matched filter to search  
021 %  
022 % Required subroutines: Use PLOT_DAMA_DATA.M to visualize data  
023 %  
024 % Notes:  
025 % 1) For n MA users DAMA-to-MA power ratio is  
026 %     SNR = (Rb_DAMA/Rc_DAMA)/(n*Rb_MA/Rc_MA)  
027 %           = (1000/10000)/(n*10000/3000000) = 30/n.  
028 %     For n = 5, SNR = 7.78dB;  
029 %  
030 % 2) This code is compute intensive!!!  
031 %  
032 % References: Digital and Analog Communication Systems by Couch p.359  
033 %  
034 % Copyright (c) 1998 by Phillip De Leon, All Rights Reserved  
035 %*****  
036  
037 clear;  
038  
039 % -----  
040 % DAMA User's Parameters  
041 %-----  
042 Fc_DAMA = 164e3; % DAMA user's carrier frequency (after shifting) in Hz  
043 main_lobe_BW = 200e3; % 200kHz main lobe BW centered on F_DAMA  
044 max_Doppler = 64e3; % maximum Doppler shift of +/- 64kHz  
045 est_tolerance = 3e3; % must estimate carrier to within +/- est_tolerance  
046  
047 %-----  
048 % Communication System Parameters  
049 %-----  
050 Fs = 800000; % output signal sampling freq. (samples/s)  
051 Fc = Fc_DAMA + 0.5*max_Doppler; % assume DAMA carrier is +50% max  
052 % Doppler shifted (for testing purposes)  
053  
054 Rb_DAMA = 1000; % data rate for DAMA (bits/s) Fs/Rb_DAMA must be integer  
055 Rc_DAMA = 100000; % chip rate for DAMA (chips/s) 100K?  
056 samples_per_bit = Fs/Rb_DAMA; % must be integer  
057 samples_per_chip = Fs/Rc_DAMA; % must be integer  
058 Rb_MA = 10*Rb_DAMA; % data rate for MA (bits/s)
```

```

059 Rc_MA = 3000000; % chip rate for MA (chips/s) 3000K
060
061 % -----
062 % Frequency Estimation Parameters
063 % -----
064 % Estimation of Fc occurs over a bandwidth of...
065 % Fc - max_Doppler - main_lobe_BW/2 < F_DAMA < Fc + max_Doppler + main_lobe_BW/2
066 % 0 <= f <= 328
067 N = 512; % length data block to take FFT over
068 window = 0; % if 0 assume rectangular window, 1 Hamming Window
069 if (window)
070   w = hamming(N); % Hamming window used in FFT
071 end;
072
073 %-----
074 % Simulation Parameters
075 %-----
076 number_of_estimates = 30; % number of frequency estimates to perform
077 % for each SNR typically 10000
078 num_FFTs = 8; % average 8 FFTs per estimate;
079 snr = [-5]; % vector of SNRs (in dB) to simulate typically from -2.2 to 4.7dB
080
081 %-----
082 % Simulation
083 %-----
084 est_data = zeros(length(snr),N/2); % malloc storage for estimation data
085
086 for m = 1:length(snr); % m is index for SNRs
087   disp(sprintf('*** SNR = %d ***',snr(m)));
088   max_pos_est_error = 0; % reset variables
089   max_neg_est_error = 0;
090   rand('seed',0); % reset uniform generator seed
091   randn('seed',0); % reset normal generator seed
092
093 for n = 1:number_of_estimates; % n is index for estimate #
094   n;
095   sum_mag_X_sqrd = zeros(N/2,1); % malloc storage for accumulation of FFT data
096   sum_mag_S_sqrd = zeros(N/2,1); % malloc storage for accumulation of FFT data
097
098 % -----
099 % Synthesize SS signal
100 %
101 % Generate digital message
102 data = round(rand(ceil(num_FFTs*N/samples_per_bit),1)); % generate data (bits) to modulate
103
104 % Digital BPSK Modulation
105 start = 0;
106 stop = samples_per_bit - 1;
107 msg = [];
108 for i = 1:length(data)
109   n = [start:stop];
110   if ("data(i)") % if 0 multiply by -1
111     msg = [msg' (-1)*sin(2*pi*n*Fc/Fs)]; % build up message signal
112   else % if 1 multiply by 1
113     msg = [msg' sin(2*pi*n*Fc/Fs)]; % build up message signal
114   end;
115   start = stop + 1;
116   stop = min((start + samples_per_bit - 1),num_FFTs*N);
117 end;
118
119 % Direct Sequence Spreading
120 PN_code = round(rand(ceil(num_FFTs*N/samples_per_chip),1));

```

```

121      start = 1;
122      stop = samples_per_chip;
123      s = [];
124      for i = 1:length(PN_code)
125          n = [start:stop];
126          if ('PN_code(i)') % if 0 multiply by -1
127              s = [s' (-1)*msg(n)'];
128          else % if 1 multiply by 1
129              s = [s' msg(n)'];
130          end;
131          start = stop + 1;
132          stop = min((start + samples_per_chip - 1),num_FFTs*N);
133      end;
134
135 % Compute FFT of BPSK-SS for Matched Filter Use
136 S = fft(s(1:N));
137 mag_S_sqrd = real(S(1:N/2).*conj(S(1:N/2)));
138 sum_mag_S_sqrd = sum_mag_S_sqrd + mag_S_sqrd;
139 avg_mag_S_sqrd = sum_mag_S_sqrd/num_FFTs;
140 MF = flipud(avg_mag_S_sqrd);
141
142 for k = 1:num_FFTs; % k is index on FFT number
143
144 % Add AWGN noise (MA users + channel noise) according to desired SNR to SS signal
145 noise = randn(N,1) .* sqrt(cov(s)/(10^(snr(m)/10))); % ./?
146 s_prime = s(1+N*(k-1):N*(k-1)+512); % Take length N buffer of samples from total.
147 r = s_prime + noise; % carrier + noise
148 r = r ./ sqrt(cov(r)); % scale to unit variance
149
150 % -----
151 % Window signal
152 % -----
153 if (~window)
154     x = r; % no window => rectangular window
155 else
156     x = w .* r; % window with preset window
157 end;
158
159 X = fft(x);
160 mag_X_sqrd = real(X(1:N/2).*conj(X(1:N/2))); % due to symmetry need only scan lower half
161 sum_mag_X_sqrd = sum_mag_X_sqrd + mag_X_sqrd;
162
163 end; % FFT loop
164
165 % -----
166 % Frequency estimation
167 % -----
168 avg_mag_X_sqrd = sum_mag_X_sqrd/num_FFTs;
169
170 % Frequency-Domain Matched Filtering
171 MF_mag_X = conv(MF,avg_mag_X_sqrd);
172
173 if (0)
174     k = find(avg_mag_X_sqrd == max(avg_mag_X_sqrd));
175     if (length(k) > 1)
176         k = median(k); % choose the median k if there are several
177     else
178         k = k;
179     end;
180 end;
181 if (1)
182     k = find(MF_mag_X == max(MF_mag_X));

```

```

183     if (length(k) > 1)
184         k = median(k); % choose the median k if there are several
185     else
186         k = k;
187     end;
188     k = k - N/2 - 1; % MF spreads range by 2
189 end;
190
191 est_data(m,k) = est_data(m,k) + 1; % tally estimate
192
193 end; % estimates loop
194 save dama_data est_data
195 end; % snr loop
196
197 % -----
198 % Save Data in MATLAB 4.x format
199 % -----
200 x = version;
201 if (x(1)==53) % running on MATLAB 5.x
202     save -v4 dama_data Rc_DAMA Fs Fc snr est_tolerance N window number_of_estimates est_data
203 else           % running on MATLAB 4.x
204     save dama_data Rc_DAMA Fs Fc snr est_tolerance N window number_of_estimates est_data avg_mag_X_sqrd
205 end;

```

mf_bpsk_ss.m

```
001 %*****  
002 % MF_BPSK  
003 %  
004 % Description: This function computes the matched filter for the PSD of the  
005 % BPSK-SS frequency-domain waveform. The matched filter coefficients are  
006 % stored in a file for use in the DAMA.M code. We may be able to use the  
007 % theoretical PSD.  
008 %  
009 % Programmer: Phillip De Leon  
010 %  
011 % Creation Date: Sept. 25, 1998  
012 % Last Revision: Oct. 24, 1998  
013 %  
014 % Version History:  
015 % 1.0 - MF for the BPSK-SS  
016 % 1.1 - New Vectorization Methods - will reduce run time.  
017 % 1.2 - Added Crash Recovery - ALL data is saved to temp.mat prior to  
018 % next major algorithm step.  
019 %  
020 % Required subroutines:  
021 %  
022 % Notes:  
023 % 1) N must be the same as in DAMA.M  
024 %  
025 % Approximate Runtime:  
026 % P200 win98 : ~22 minutes  
027 % Mac : Not performed  
028 % Pxx Linux : Not performed  
029 %  
030 % References:  
031 %  
032 % Copyright (c) 1998 by Phillip De Leon, All Rights Reserved  
033 %*****  
034 ON = 1; % Used to turn options on or off.  
035 OFF = 0;  
036 ERR = 0; % Set ERR to level achieved before crash - results  
037 % are saved in temp.mat  
038  
039 if ERR == 0 disp('Operation Level One:')
```

040 % -----
041 % Code Parameters
042 % -----
043 spreading_opt = ON; % ON to spread - OFF to just BPSK modulate.
044 window = OFF; % if OFF assume rectangular window, ON Hamming Window
045
046 % -----
047 % DAMA User's Parameters
048 % -----
049 Fc_DAMA = 200e3; % DAMA user's carrier frequency (after shifting) in Hz
050 % Set to middle of DAMA frequency range.
051
052 % -----
053 % Communication System Parameters
054 % -----
055 Fs = 800000; % output signal sampling freq. (samples/s)
056 Fc = Fc_DAMA; % assume unshifted DAMA carrier
057
058 Rb_DAMA = 1000; % data rate for DAMA (bits/s) Fs/Rb_DAMA must be integer
059 Rc_DAMA = 50000; % chip rate for DAMA (chips/s) 100K?
060 if rem(Fs,Rb_DAMA) ~= 0

```

061     error('samples_per_bit MUST be an integer!!') % Check to make sure that the
062                                     % samples per bit is integer.
063 end
064 if rem(Fs,Rc_DAMA) ~= 0
065     error('samples_per_chip MUST be an integer!!') % Check to make sure that the
066                                     % samples per chip is integer.
067 end
068 samples_per_bit = Fs/Rb_DAMA;    % must be integer
069 samples_per_chip = Fs/Rc_DAMA;  % must be integer
070
071 % -----
072 % Frequency Estimation Parameters
073 % -----
074 % Estimation of Fc occurs over a bandwidth of...
075 % Fc - max_Doppler - main_lobe_BW/2 < F_DAMA < Fc + max_Doppler + main_lobe_BW/2
076 % 0 <= f <= 328
077 N = 512; % length data block to take FFT over
078
079 if (window)
080     w = hamming(N); % Hamming window used in FFT
081 else
082     w = ones(N,1);
083 end
084
085 % -----
086 % Averaging Parameters
087 % -----
088 num_FFTs = 5000;    % Average iterations - higher -> better resolution
089
090 save temp           % End of level One - save work so far to temp.
091 end
092
093 if ERR == 1 | ERR == 0 % Check Error Level.
094 disp('Operation Level Two:')
095 % -----
096 % Generate digital message
097 % -----
098 M = num_FFTs*N*Rb_DAMA/Fs; % number of bits to take PSD over
099
100 disp('Preallocating...')
101 msg = zeros(M*samples_per_bit,1); % Preallocate msg signal
102 data = msg;                   % Preallocate data signal (-1,+1)
103 s = msg;                     % Preallocate s - see variable description.
104 disp('Done Preallocating...')
105
106 msg = sin(2*pi*(0:M*samples_per_bit-1)*Fc/Fs); % BIG vector - not for the weak!!
107 msg = msg'; data =
108 filter(ones(samples_per_bit,1),1,upsamp((-1).^(round(rand(M,1)))),samples_per_bit));
109                                     % Previous line generates a data vector with each
110                                     % bit represented by samples_per_bit number of
111                                     % samples.
112 msg = msg.*data;
113                                     % data vector is multiplied by the signal vector
114                                     % to generate the message vector - BPSK modulated.
115 save temp           % End of Level Two - save work so far to temp.
116 end
117
118 if ERR == 2 | ERR == 1 | ERR == 0; disp('Operation Level Three:')
119
120 % -----
121 % Spread BPSK Signal
122 % -----

```

```

123 if spreading_opt      % If spreading option is on then spread signal.
124 PN_code = filter(ones(samples_per_chip,1),1, ...
125     upsamp((-1).^(round(rand(ceil(num_FFTs*N/samples_per_chip),1)))),...
126     samples_per_chip);
127 s = msg.*PN_code;    % s is the BPSK modulated spread spectrum signal.
128 end
129
130 % Compute FFT of BPSK-SS for Matched Filter Use from averages.
131
132 % h = waitbar2(0,'Generating Matched Filter Frequency Response...');
133 start = 1;
134 stop = N;
135
136 save temp        % End of Level Three - save work so far to temp.
137 end
138
139 if ERR == 3 | ERR == 2 | ERR == 1 | ERR == 0
140
141 disp('Operation Level Four:')
142
143 sum_mag_S_sqrd = zeros(N/2,1);
144 for l = 1:num_FFTs          % Generate averaged mag-squared
145     % waitbar2(l/num_FFTs);    % data from NN block FFTs.
146     l
147     n = [start:stop];
148     temp = w.*s(n);
149     S = fft(temp);
150     mag_S_sqrd = real(S(1:N/2).*conj(S(1:N/2)));
151     sum_mag_S_sqrd = sum_mag_S_sqrd + mag_S_sqrd;
152     start = stop + 1;
153     stop = stop + N;
154 end
155
156 % close(h)
157 avg_mag_S_sqrd = sum_mag_S_sqrd/num_FFTs;    % average.
158 MF = flipud(avg_mag_S_sqrd);                  % "time reverse" data to prepare
159 N1 = Fc*N/Fs;                                % for dama_mf code.
160                                         % Record index of carrier frequency.
161
162 end
163
164 if window
165     wtype = ['ha'];
166 else
167     wtype = ['re'];
168 end fname = ['pg' int2str(Rc_DAMA/Rb_DAMA) wtype];
169
170 mver = version if (mver(1) == '5')
171     eval(['save -v4 ' fname ' MF N1 N'])
172 else
173     eval(['save ' fname ' MF N1 N'])
174 end
175
176
177 % -----
178 %
179 % Variable Description
180 %
181 % Name           Description           Purpose
182 %
183 % ON            Program Control       Used to turn ON/OFF Program Options
184 % OFF           Program Control       Used to turn ON/OFF Program Options

```

```

185 % ERR           Program Control    Used in disaster recovery
186 % spreading_opt Program Control    Used to turn ON/OFF spreading
187 % window          Program Control    Used to turn ON/OFF windowing
188 % Fc_DAMA         Program Parameter DAMA Carrier Frequency - for the
189 %                  purposes of this program it is used
190 %                  to set the reference index.
191 % Fs              Program Parameter Sampling Frequency
192 % Fc              Program Parameter Carrier Frequency - in this program
193 %                  it is set to the value of Fc_DAMA.
194 % Rb_DAMA         Program Parameter Bit Rate
195 % Rc_DAMA         Program Parameter Chip Rate
196 % samples_per_bit
197 %                  Calculated Parameter Each bit is represented by # of samples
198 % samples_per_chip
199 %                  Calculated Parameter Each chip is represented by # of
200 %                  samples.
201 % N               Program Parameter Length of FFT block
202 % w               Calculated Parameter window coefficients
203 % num_FFTs        Program Parameter Number of Iterations
204 % M               Calculated Parameter The number of bits required in the data
205 %                  vector.
206 % msg             Calculated Vector The actual sinusoidal signal - later it is
207 %                  BPSK modulated.
208 % data            Calculated Vector Binary data stored as (-1,1)
209 % s               Calculated Vector BPSK SS carrier
210 % PN_code         Calculated Vector The psuedo-random noise vector to cause
211 %                  spreading.
212 % start           Program Parameter Used in indexing
213 % stop            Program Parameter Used in indexing
214 % l               Program Control   Used in FOR loop
215 % h               Program Control   Handle to figure
216 % sum_mag_S_sqrd
217 %                  Calculated Vector Holds sum of magnitude squared data from FFT.
218 % S               Calculated Vector Holds FFT data for current iteration
219 % mag_S_sqrd      Calculated Vector Holds magnitude squared data from FFT for
220 %                  current iteration.
221 % avg_mag_S_sqrd
222 %                  Calculated Vector Holds the average mag_S_sqrd vector.
223 % MF              Calculated Vector Frequency reversed - think of it as time
224 %                  reversed data.
225 % wtype           Program Control   Type of window employed for inclusion into
226 %                  file name.
227 % fname           Program Control   Filename
228 % mver            Program Control   mver(1) holds matlab version.
229 %
230 %-----
```

plot_dama_data.m

```
001 %*****  
002 % PLOT_DAMA_DATA plot_dama_data  
003 %  
004 % Description: This function provides a variety of tools for DAMA data  
005 % visualization using the file saved in DAMA.M  
006 %  
007 % Call Syntax: plot_dama_data  
008 %  
009 % Programmer: Phillip De Leon  
010 %  
011 % Creation Date: December 31, 1995  
012 % Last Revision: July 1, 1997  
013 %  
014 % Required subroutines:  
015 %  
016 % Notes: Assume workspace (DAMA_DATA.MAT) has already been loaded.  
017 %  
018 % References:  
019 %  
020 % Copyright (c) 1998 by Phillip De Leon, All Rights Reserved  
021 %*****  
022  
023 %clg;  
024  
025 selection = menu('Select a Plot',...  
026     'Estimation Ranges',...    % 1)  
027     'Estimation Histograms',... % 2)  
028     'Estimation Accuracy',...  % 3)  
029     'Exit');  
030  
031 % -----  
032 % Estimation range  
033 % -----  
034 if (selection == 1)  
035     i = find(est_data(1,:));    % find non-zero elements  
036     max_neg_est_error = ((min(i)-1) * Fs / N) - Fc;  % 1 <= i <= N but need 0 <= i <= N-1  
037     max_pos_est_error = ((max(i)-1) * Fs / N) - Fc;  
038     plot([snr(1) snr(1)], [max_neg_est_error max_pos_est_error]);  
039     hold on  
040     for m = 2:length(snr)  
041         i = find(est_data(m,:));    % find non-zero elements  
042         max_neg_est_error = ((min(i)-1) * Fs / N) - Fc;  
043         max_pos_est_error = ((max(i)-1) * Fs / N) - Fc;  
044         plot([snr(m) snr(m)], [max_neg_est_error max_pos_est_error]);  
045     end;  
046     plot([-100 100],[est_tolerance est_tolerance],':')  
047     plot([-100 100],[-est_tolerance -est_tolerance],':')  
048     hold off;  
049     ylabel('Estimation error (Hz)');  
050     xlabel('SNR (dB)');  
051     if (window)  
052         title(['Frequency Estimation Ranges (Rc=',sprintf('%d',Rc_DAMA), ...  
053                 ', N=',sprintf('%d', N),', Hamming window')]);  
054     else  
055         title(['Frequency Estimation Ranges (Rc=',sprintf('%d',Rc_DAMA), ...  
056                 ', N=',sprintf('%d', N),', Rectangular window')]);  
057     end;  
058     [i,j] = find(est_data);  
059     Ymin = min(j);  
060     Ymax = max(j);
```

```

061     axis([(snr(1)-10) (snr(length(snr))+10) 1.1*min(Ymin,-est_tolerance) ...
062             1.1*max(Ymax,est_tolerance)]);
063
064 % -----
065 % Estimation histogram
066 % -----
067 elseif (selection == 2)
068     for m = 1:length(snr)
069         bar([0:N/2-1].*(Fs/N),est_data(m,1:N/2));
070         hold on;
071         plot([0:N/2-1].*(Fs/N),est_data(m,1:N/2),'+');
072         plot([Fc Fc],[0 1.1*max(est_data(m,1:N/2))], 'b:');
073         plot([Fc+est_tolerance Fc+est_tolerance],[0 1.1*max(est_data(m,1:N/2))], 'r:');
074         plot([Fc-est_tolerance Fc-est_tolerance],[0 1.1*max(est_data(m,1:N/2))], 'r:');
075         hold off;
076         ylabel('Occurrences')
077         xlabel('Estimation Frequency (Hz)')
078         if (window)
079             title(['Frequency Estimation for SNR of ',sprintf('%g', snr(m)), ...
080                   'dB (Rc=',sprintf('%d',Rc_DAMA),', N=',sprintf('%d',N),', Hamming window')]);
081         else
082             % title(['Frequency Estimation for SNR of ',sprintf('%g', snr(m)), ...
083             % 'dB (Rc=',sprintf('%d',Rc_DAMA),', N=',sprintf('%d',N),', Rectangle window')]);
084             title(['Frequency Estimation for SNR of ',sprintf('%g', snr(m)), ...
085                   'dB']);
086         end;
087         axis([0.95*(Fc-est_tolerance) 1.05*(Fc+est_tolerance) 0 1.1*max(est_data(m,1:N/2))]);
088         disp('Hit any key to plot next histogram... ');
089         pause;
090     end;
091
092 % -----
093 % Statistical Accuracy
094 % -----
095 elseif (selection == 3)
096     lower_index_bound = ceil(N*(Fc-est_tolerance)/Fs)+1 % 1 <= lower_index_bound <= N
097 %     lower_index_bound = 1;
098     upper_index_bound = floor(N*(Fc+est_tolerance)/Fs)+1 % 1 <= upper_index_bound <= N
099 %     upper_index_bound = 211;
100
101    pause
102    accuracy = zeros(length(snr),1);
103    for m = 1:length(snr)
104        number_accurate_estimates=sum(est_data(m,lower_index_bound:upper_index_bound));
105        accuracy(m) = number_accurate_estimates / number_of_estimates;
106    end;
107    plot(snr,accuracy,'k')
108    xlabel('SNR (dB)');
109    ylabel('Proportion of Estimates in Range');
110    title('Estimation Accuracy');
111    grid;
112 else
113     return;
114 end;
115 % plot_dama_data

```

wsands4.m

```
001 % White Sands Data Collection Code ver 4.0
002 % Started on 6/26/98
003 % Fill in comments. ---> Still need to comment
004 %
005 % Signal Vector Descriptions:
006 % 1: Rb = 1000 bits/sec   Rc = 100000 chips/sec
007 % 2: Rb = 1000 bits/sec   Rc = 20000 chips/sec
008 % 3: Rb = 1000 bits/sec   Rc = 10000 chips/sec
009 % 4: Rb = 1000 bits/sec   Rc = 40000 chips/sec
010 %
011 % Revision History:
012 %
013 % Ver 1.0: Baseline, one file generated per user input
014 % Ver 2.0: Four "useful" files generated at once, no user
015 %           input. Some code cleaned up (vectorized).
016 % Ver 3.0: More vectorization, utilizes MATLAB's filter
017 %           function. Vectors are pre-allocated.
018 % Ver 3.01: Changed code so that # of bits may be individually
019 %           selected.
020 % Ver 4.0: To allow for larger data file generation, a menu
021 %           system is implemented so that only one data file
022 %           is generated. This compensates for memory
023 %           problems. Edit file for different spread/data
024 %           rates.
025 % Ver 4.01: "2047" NASA PN sequence took to long to generate
026 %           so the sequence is generated offline for faster
027 %           performance. This restricts resulting binary file
028 %           length to 2meg.
029 %
030 % Last Update: 7/13/98
031 % Written by: Bradley James Scaife
032 %*****
033 timer = cputime
034 preamble_length = 5000;
035
036 choice = menu('Please Select File to be
037 Generated:','1k_100','1k_20','1k_10','1k_40');
038
039 if choice == 1
040 % Generate file one
041 %*****
042
043 % Message One Parameters
044 %*****
045 %msg_len1 = input('Enter Message Length One (bits): ');
046 msg_len1 = 20e3; %Restricted value - only change if spread vector code is altered.
047 Rbi = 1000; Rci = 100000;
048
049 % Calculated Parameters: Message One
050 %*****
051 spread_factor1 = ceil(Rci/Rbi); % Set as an integer.
052
053 % Preallocate MSG1 vector
054 %*****
055 msg1_init = zeros(msg_len1,1); msg1 =
056 zeros(msg_len1*spread_factor1,1);
057 %spreading_vector1 = zeros(msg_len1*spread_factor1,1);
058
059 % MSG 1 and Spreading Vector Generation
060 %*****
```

```

061 rand('seed',0);
062
063 msg1_init = round(rand(msg1_len1,1));
064
065 msg1 = upsamp(msg1_init,spread_factor1)';
066 msg1 = filter(ones(spread_factor1,1),1,msg1);
067 clear msg1_init
068
069 load svec
070
071 % Generate s1
072 %*****
073 disp('Generating s1... ')
074
075 s1 = 255*xor(msg1,spreading_vector)';
076
077 % Binary File Generation
078 %*****
079 disp('Generating binary file... ')
080 disp('Inserting preamble... ')
081 disp('Inserting signal vector... ')
082
083 data_out1 = [zeros(1,preamble_length) s1];
084 fid = fopen('1k_100.bin','wb');
085 fwrite(fid,data_out1,'int8');
086 fclose(fid);
087
088 elseif choice == 2
089
090 % Generate File 2: Per definition in header above.
091 %*****
092 %msg_len2 = input('Enter Message Length Two (bits): ');
093 msg_len2 = 100e3; %Restricted value - only change if spread vector code is altered.
094 Rb2 = 1000; Rc2 = 20000;
095
096 % Calculated Parameters
097 %*****
098 spread_factor2 = ceil(Rc2/Rb2);
099
100 % Preallocate MSG2 vectors
101 %*****
102 msg2_init = zeros(msg2_len2,1);
103 msg2 = zeros(msg2_len2*spread_factor2,1);
104 %spreading_vector2 = zeros(msg2_len2*spread_factor2,1);
105
106 % Message and Spreading Vector Generation
107 %*****
108 msg2_init = round(rand(msg2_len2,1));
109 msg2 = upsamp(msg2_init,spread_factor2)';
110 msg2 = filter(ones(spread_factor2,1),1,msg2);
111 clear msg2_init
112
113 load svec
114
115 % Generate s2
116 %*****
117 disp('Generating s2... ')
118
119 s2 = 255*xor(msg2,spreading_vector)';
120
121 % Binary File Generation
122 %*****

```

```

123 disp('Generating binary file...')
124 disp('Inserting preamble...')
125 disp('Inserting signal vector...')
126
127 data_out2 = [zeros(1,preamble_length) s2];
128
129 fid = fopen('1k_20.bin','wb');
130 fwrite(fid,data_out2,'int8');
131 fclose(fid);
132
133
134 elseif choice == 3
135 % Generate File 3: Per definition in header above.
136 %*****
137 %msg_len3 = input('Enter Message Length Three (bits): ');
138 msg_len3 = 200e3; %Restricted value - only change if spread vector code is altered.
139 Rb3 = 1000; Rc3 = 10000;
140
141 % Calculated Parameters
142 %*****
143 spread_factor3 = ceil(Rc3/Rb3);
144
145 % Preallocate MSG3 vectors
146 %*****
147 msg3_init = zeros(msg_len3,1);
148 msg3 = zeros(msg_len3*spread_factor3,1);
149 spreading_vector3 = zeros(msg_len3*spread_factor3,1);
150
151 % Message and Spreading Vector Generation
152 %*****
153 msg3_init = round(rand(msg_len3,1));
154 msg3 =upsamp(msg3_init,spread_factor3)';
155 msg3 = filter(ones(spread_factor3,1),1,msg3);
156 clear msg3_init
157
158 load svec
159
160 % Generate s3
161 %*****
162 disp('Generating s3...')
163
164 s3 = 255*xor(msg3,spreading_vector)';
165
166 % Binary File Generation
167 %*****
168 disp('Generating binary file...')
169 disp('Inserting preamble...')
170 disp('Inserting signal vector...')
171
172 data_out3 = [zeros(1,preamble_length) s3];
173
174 fid = fopen('1k_10.bin','wb');
175 fwrite(fid,data_out3,'int8');
176 fclose(fid);
177
178 else
179 % Generate File 4: Per definition in header above.
180 %*****
181 %msg_len4 = input('Enter Message Length Four (bits): ');
182 msg_len4 = 50e3; %Restricted value - only change if spread vector code is altered.
183 Rb4 = 1000; Rc4 = 40000;
184

```

```

185 % Calculated Parameters
186 %*****
187 spread_factor4 = ceil(Rc4/Rb4);
188
189 % Preallocate MSG4 vectors
190 %*****
191 msg4_init = zeros(msg_len4,1);
192
193 msg4 = zeros(msg_len4*spread_factor4,1);
194 % spreading_vector4 = zeros(msg_len4*spread_factor4,1);
195
196 % Message and Spreading Vector Generation
197 %*****
198 msg4_init = round(rand(msg_len4,1));
199
200 msg4 = upsamp(msg4_init,spread_factor4)';
201
202 msg4 = filter(ones(spread_factor4,1),1,msg4);
203 clear msg4_init
204
205 load svec
206
207 % Generate s4
208 %*****
209 disp('Generating s4... ')
210
211 s4 = 255*xor(msg4,spreading_vector)';
212
213 % Binary File Generation
214 %*****
215 disp('Generating binary file... ')
216 disp('Inserting preamble... ')
217 disp('Inserting signal vector... ')
218
219 data_out4 = [zeros(1,preamble_length) s4];
220
221 fid = fopen('1k_40.bin','wb');
222 fwrite(fid,data_out4,'int8');
223 fclose(fid);
224
225 end
226
227 disp('All Done : ) ')
228 run_time = cputime - timer

```

cap_mf.m

```
001 % Title:  
002 %   cap_mf.m  
003 %  
004 % Purpose:  
005 %   The purpose of this code is to generate estimation data similar to  
006 %   dama_mf with the exception that this code is tailored to use data  
007 %   captured during the White Sands Complex (WSC) experiment.  
008 %  
009 % Revision:  
010 %   1.0 -- 11/20/98  
011 %  
012 % Revision History:  
013 %   none => baseline  
014 %  
015 % Author:  
016 %   Brad Scaife  
017 %  
018 % Notes:  
019 %   This code requires as input data that has been prepared by wsco_d2.m which  
020 %   prepares the captured data and generates a matrix of usable vectors (see  
021 %   wsco_d2 comments for more information). This code also relies on an index  
022 %   matrix that is generated by in_prp.m. In both cases these files have already been  
023 %   prepared and may simply be loaded following prescribed naming conventions. In  
024 %   the event that these data files have been lost though it was thought helpful to  
025 %   comment on how to regenerate them.  
026 %  
027 % Average runtime:  
028 %   ~ 8 minutes  
029 %  
030 %*****  
031 clear all;  
032  
033 ON = 1; OFF = 0;  
034  
035 % Code Options  
036 %:::::::::::  
037 window = OFF;           % Use to toggle use of hamming window  
038 matched_filter = ON;    % Use to toggle use of matched filter  
039 insight = OFF;          % Use to gain instantaneous frequency estimation  
040  
041 % Code Parameters  
042 %:::::::::::  
043 Fs = 100e6;             % DO NOT CHANGE - this was the sample rate used.  
044 %snr = 1.5               % As per dama_mf definition - not wsc definition.  
045 est_tolerance = 3e3;% +/- range of acceptable error in Hz.  
046 N = 65536;              % FFT block length.  
047 number_of_estimates = 1000; % # of estimation attempts. Don't modify!!!  
048 fft_avg = 8;             % Number of FFT's to average over - changing this will  
049                           % require changes to code mentioned in documentation above  
050                           % as well as the rerunning of this code.  
051  
052 alpha = 2;               % Constant used to obtain consistent data points - does not  
053                           % add to statistical meaning nor detract from it.  
054 f_lo = 32.50e6;  
055 f_hi = 32.90e6;  
056  
057 lower_bound_index = ceil(f_lo*N/Fs)+1;  
058  
059 starting_bound = lower_bound_index;  
060
```

```

061 upper_bound_index = floor(f_hi*N/Fs)+1;
062 Rb_DAMA = 1e3;
063
064 % Program Flow
065 %:::::::::::;;
066 if window
067     wtype = ['h'];
068 else
069     wtype = ['r'];
070 end
071
072 menu_sel = menu('Select Data to Process:',...
073     'SF=10, SNR1, F2',...           % Selection 1
074     'SF=10, SNR1, F3',...           % Selection 2
075     'SF=10, SNR2, F1',...           % Selection 3
076     'SF=10, SNR2, F2',...           % Selection 4
077     'SF=10, SNR2, F3',...           % Selection 5
078     'SF=10, SNR2, F5',...           % Selection 6
079     'SF=10, SNR3, F2',...           % Selection 7
080     'SF=20, SNR2, F3')            % Selection 8
081
082 if menu_sel == 1
083     data_filename = ['c:\research\code\matlab\whites^2\wscmat^1\snr1_f2' wtype];
084     eval(['load ' data_filename]);
085     num_col = size(x,2);
086     index_filename = ['c:\research\code\matlab\whites^2\cmb8_ ' int2str(num_col)];
087     eval(['load ' index_filename]);
088     iterations = choose(num_col,8);
089     %snr = -5;
090     snr = 40;
091     Rc_DAMA = 10e3;
092     Fc = 32.65e6;
093
094 elseif menu_sel == 2
095     data_filename = ['c:\research\code\matlab\whites^2\wscmat^1\snr1_f3' wtype];
096     eval(['load ' data_filename]);
097     num_col = size(x,2);
098     index_filename = ['c:\research\code\matlab\whites^2\cmb8_ ' int2str(num_col)];
099     eval(['load ' index_filename]);
100    iterations = choose(num_col,8);
101    %snr = -5;
102    snr = 40;
103    Rc_DAMA = 10e3;
104    Fc = 32.70e6;
105
106
107 elseif menu_sel == 3
108     data_filename = ['c:\research\code\matlab\whites^2\wscmat^1\snr2_f1' wtype];
109     eval(['load ' data_filename]);
110     num_col = size(x,2);
111     index_filename = ['c:\research\code\matlab\whites^2\cmb8_ ' int2str(num_col)];
112     eval(['load ' index_filename]);
113     iterations = choose(num_col,8);
114     %snr = 1.5;
115     snr = 45;
116     Rc_DAMA = 10e3;
117     Fc = 32.6e6;
118
119
120 elseif menu_sel == 4
121     data_filename = ['c:\research\code\matlab\whites^2\wscmat^1\snr2_f2' wtype];
122     eval(['load ' data_filename]);

```

```

123 num_col = size(x,2);
124 index_filename = ['c:\research\code\matlab\whites^2\cmb8_ int2str(num_col)];
125 eval(['load ' index_filename]);
126 iterations = choose(num_col,8);
127 % snr = 1.5;
128 snr = 45;
129 Rc_DAMA = 10e3;
130 Fc = 32.65e6;
131
132
133 elseif menu_sel == 5
134 data_filename = ['c:\research\code\matlab\whites^2\wscmat^1\snr2_f3' wtype];
135 eval(['load ' data_filename]);
136 num_col = size(x,2);
137 index_filename = ['c:\research\code\matlab\whites^2\cmb8_ int2str(num_col)];
138 eval(['load ' index_filename]);
139 iterations = choose(num_col,8);
140 % snr = 1.5;
141 snr = 45;
142 Rc_DAMA = 10e3;
143 Fc = 32.70e6;
144
145
146 elseif menu_sel == 6
147 data_filename = ['c:\research\code\matlab\whites^2\wscmat^1\snr2_f5' wtype];
148 eval(['load ' data_filename]);
149 num_col = size(x,2);
150 index_filename = ['c:\research\code\matlab\whites^2\cmb8_ int2str(num_col)];
151 eval(['load ' index_filename]);
152 iterations = choose(num_col,8);
153 %snr = 1.5;
154 snr = 45;
155 Rc_DAMA = 10e3;
156 Fc = 32.80e6;
157
158
159 elseif menu_sel == 7
160 data_filename = ['c:\research\code\matlab\whites^2\wscmat^1\snr3_f2' wtype];
161 eval(['load ' data_filename]);
162 num_col = size(x,2);
163 index_filename = ['c:\research\code\matlab\whites^2\cmb8_ int2str(num_col)];
164 eval(['load ' index_filename]);
165 iterations = choose(num_col,8);
166 %snr = 2.2;
167 snr = 50;
168 Rc_DAMA = 10e3;
169 Fc = 32.65e6;
170
171
172 else
173 data_filename = ['c:\research\code\matlab\whites^2\wscmat^2\snr2_f3' wtype];
174 eval(['load ' data_filename]);
175 num_col = size(x,2);
176 index_filename = ['c:\research\code\matlab\whites^2\cmb8_ int2str(num_col)];
177 eval(['load ' index_filename]);
178 iterations = choose(num_col,8);
179 snr = 1.5;
180 Rc_DAMA = 10e3;
181 Fc = 32.70e6;
182
183 end
184
```

```

185 % Some Corrections for code operation
186 %:::::::::::;;
187 if iterations > 1000
188     iterations = 1000;
189 end
190
191 if iterations == 1000
192     alpha = 1;
193 else
194     alpha = 2;
195 end
196
197 % Load Matched Filter
198 %::::::::::;;
199 if matched_filter
200     if window
201         wtype2 = ['ha'];
202     else
203         wtype2 = ['re'];
204     end
205
206     fname = ['pg' int2str(Rc_DAMA/Rb_DAMA) wtype2];
207     eval(['load c:\research\code\matlab\damane\1\' fname])
208 end
209
210 % Process Begin
211 %::::::::::;;
212 N = 65536; % Reset N after loading matched filter - required.
213 % Preallocation
214 %::::::::::;;
215 X = zeros(N,fft_avg);
216 %X = zeros(N/2,fft_avg);
217 X_mag_squared = zeros(length(lower_bound_index:upper_bound_index),fft_avg);
218 %X_mag_squared = zeros(N/2,fft_avg);
219 X_avg = zeros(length(lower_bound_index:upper_bound_index),fft_avg);
220 %X_avg = zeros(N/2,fft_avg);
221 est_data = zeros(length(snr),length(lower_bound_index:upper_bound_index));
222 %est_data = zeros(length(snr),N/2);
223
224 % Begin FFT Estimation Process
225 %::::::::::;;
226 for n = 1:iterations
227     n
228     temp = x(:,index(n,:)); % Select current index pattern.
229     X = fft(temp); % Perform 8 FFT's
230     X = X(lower_bound_index:upper_bound_index,:); % Restrict to search bound
231     %X = Xt(1:N/2,:);
232     X_mag_squared = (X.*conj(X)).';
233     X_avg = (sum(X_mag_squared) ./ fft_avg).'; % Sum and average.
234
235     if matched_filter
236         MF_mag_X = conv(MF,X_avg);
237     end
238
239     k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg))));;
240     if insight
241         disp('Frequency found to be:')
242         (k+20791)*100e6/64e3
243         pause(10)
244     end
245     if matched_filter
246         k = find(MF_mag_X(1:length(MF_mag_X)) == max(MF_mag_X(1:length(MF_mag_X))));;

```

```

247
248     if (length(k) > 1)
249         k = median(k);
250     end
251
252     k = k - N1;
253 else
254     k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg))));
```

.

```

255
256     if (length(k) > 1)
257         k = median(k);
258     end
259
260 end
261
262     est_data(1,k) = est_data(1,k) + 1; % Increment bin
263
264 end
265
266 est_data(1,:) = est_data(1,:).* alpha; % Sponge data to look like a larger estimate
267
268 % -----
269 % Save Data in MATLAB 4.x format
270 % -----
271 x = version;
272 if (x(1)=='5') % running on MATLAB 5.x
273     save -v4 cap_dat Fc Fs N N1 est_data est_tolerance number_of_estimates ...
274             starting_bound snr window wtype2
275 else % running on MATLAB 4.x
276     save cap_dat Fc Fs N N1 est_data est_tolerance number_of_estimates ...
277             starting_bound snr window wtype2
278 end;
```

wsco_d2.m

```
001 % DAMA Test Signal Analysis
002 %
003 % Purpose:
004 %   This code is to be used as driver code for wsco.m. This code
005 %   converts captured whitesands files and converts them into
006 %   usable length N blocks. Not entirely automated. Please read
007 %   info in wsco.m for details on project purpose.
008 %
009 % Input:
010 %   This code requires the DAMA test files named as s1,s2,...sn.
011 %
012 % Output:
013 %   Undecided at this point.
014 %
015 % Revision History:
016 %
017 %   ver 1.0: baseline
018 %   ver 2.0: Changed scope of code. This program is now to
019 %             to be used to drive other code only.
020 %   ver 2.5: Restructured the format data is to be saved in.
021 %
022 %
023 % Current Version - Date:
024 %   2.5 - 8/4/98
025 %
026 % Author:
027 %   Brad Scaife
028 %
029 % Date:
030 %   7/23/98
031 %
032 % Notes:
033 %   Remember to change the save filename at the end of the code. Sorry
034 %       got to do a little work to run this one.
035 %*****
036 clear all
037 ON = 1;
038 OFF = 0;
039 window = OFF;
040
041 % Change to working directory
042 %*****
043 cd c:\research\data\damaus~1\1k_10\snr2\f3
044
045 % Code Options
046 %*****
047 % capture_option = 0;      % 0 for 50ksample data else 1 (for 100k)
048
049 % Code Parameters
050 %*****
051 num_files = 6;          % 6- for snr1_f3; 9- for snr2_f3; 12- for snr3_f2
052 half = 65536;
053 short_cap = 50002;
054 long_cap = 100002;
055
056 load s1;
057 if length(s1) == long_cap
058     capture_option = 1;
059 else
060     capture_option = 0;
```

```

061 end
062 clear s1;
063 pack
064
065
066
067
068
069 % Load raw signals
070 %*****
071
072 k = 0;
073
074 h = waitbar2(0,'Loading files...');
075
076 while 1
077     k = k+1;
078     waitbar2(k/num_files);
079     sk = ['s' int2str(k)];
080     filename = sk;
081     if ~exist(filename), break, end
082     eval(['load ' filename])
083     end
084 close(h)
085
086 % Preallocation
087 %*****
088 x = zeros(half,(1+capture_option)*num_files);
089
090
091 cd c:\research\code\matlab\whites^2
092
093 if capture_option
094     h = waitbar2(0,'Parsing 100 kSample Files...');
095     index = 1;
096
097     if window
098         W = hamm(half);
099     else
100         W = ones(half,1);
101     end
102
103     for k = 1:2:2*num_files
104         waitbar2(index/num_files);
105         varname = ['s' int2str(index)];
106         temp = eval([varname]);
107         index = index +1;
108         x(:,k) = temp(1:half).*W;      % For others
109         x(:,k+1) = temp(length(temp)-half+1:length(temp)).*W; % For others
110     end
111
112     close(h);
113 else
114     h = waitbar2(0,'Parsing 50 kSample Files...');
115
116     if window
117         W = hamm(length(s1));
118     else
119         W = ones(length(s1),1);
120     end
121
122     for k = 1:num_files

```

```
123     waitbar2(k/num_files);
124     varname = ['s' int2str(k)];
125     temp = eval([varname]);
126     temp = temp.*W;
127     temp = [temp;zeros(half - length(temp),1)];
128     x(:,k) = temp;
129 end
130 close(h);
131 end
132
133 cd c:\research\code\matlab\whites^2\wscmat^2
134
135 mlver = version;
136 if (mlver(1) == '5')          % Running on MATLAB 5.x
137     save -v4 snr2_f3r x
138 else
139     save snr2_f3r x
140 end
141
142 cd c:\research\code\matlab\whites^2
143 %clear all
```

prepcap2.m

```
001 % Title:  
002 %   prepcap.m  
003 %  
004 % Purpose:  
005 %   The purpose of this code is to prepare DAMA captured signals for  
006 %   comparison against simulated results.  
007 %  
008 % Revision:  
009 %       1.0 -- 7/26/98  
010 %       1.5 -- 8/2/98  
011 %       2.0 -- 11/2/98  
012 %  
013 % Revision History:  
014 %       none  
015 %       1.5 -- Altered code to accept new signal matrix generated  
016 %           in wsco_d.m  
017 %       2.0 -- Employing Spectral Matched Filter  
018 %  
019 % Author:  
020 %   Brad Scaife  
021 %  
022 % Date:  
023 %   7/26/98  
024 %  
025 % Notes:  
026 %   This code requires input signals matrices that have been prepared  
027 %   by wsco_d.m and vector sequence matrices prepared by goob.m. Due to  
028 %   limited test set, the code makes some approximations that may or may  
029 %   not be valid. Therefore, this code should not be considered in any  
030 %   way a proof but rather a "ballpark" type of justification: Could it  
031 %   work in the real world?  
032 %  
033 %*****  
034 clear all  
035 ON = 1;  
036 OFF = 0;  
037  
038 % Code Parameters  
039 %*****  
040 snr = 1:7;          % SNR Ranges from So/N estimate.  
041 Fc = 32.7e6;        % Measured DAMA carrier - from WSC.  
042 Fs = 100e6;         % Burr-Brown Sampling Rate.  
043 est_tolerance = 3e3; % +/- acceptable error.  
044 N = 65536;          % FFT block length.  
045 number_of_estimates = 1000; % # of estimation attempts  
046 fft_avg = 8;         % Number of FFT's to perform estimation over.  
047 alpha = 2;           % Correction to number of estimates.  
048 insight = 0;  
049 lower_bound_index = 21291; % Index of lower search bound.  
050 upper_bound_index = 21504; % Index of upper search bound.  
051 starting_bound = lower_bound_index;  
052 window = OFF;  
053 matched_filter = ON;  
054  
055 % Preallocation  
056 %*****  
057 X = zeros(N,fft_avg);  
058  
059 X_mag_squared = zeros(length(lower_bound_index:upper_bound_index),fft_avg);  
060
```

```

061 X_avg = zeros(length(lower_bound_index:upper_bound_index),fft_avg);
062
063 est_data = zeros(length(snr),length(lower_bound_index:upper_bound_index));
064
065 if window
066     load c:\research\code\matlab\damane^1\pg10ha
067 else
068     load c:\research\code\matlab\damane^1\pg10re
069 end
070
071
072 % Load SNR1_F2 Data Matrix (65536x12) and Combination Matrix (495x8)
073 %***** ****
074 load c:\research\code\matlab\whites^2\snr1_f2 load
075 c:\research\code\matlab\whites^2\cmb8_12 h =
076 waitbar2(0,'Formulating SNR1 Estimation for f2...');
077
078 for n = 1:495
079
080     index = cmbo3;      % Load unique index pattern matrix.
081     waitbar2(n/495);
082
083     temp = x(:,index(n,:)); % Select current index pattern.
084     X = fft(temp);        % Perform 8 FFT's
085     X = X(lower_bound_index:upper_bound_index,:);    % Restrict to search bound
086     X_mag_squared = (X.*conj(X)).';
087     X_avg = (sum(X_mag_squared) ./ fft_avg).'; % Sum and average.
088
089     k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg)))); % Find the maximum value in the row
090     if insight
091         disp('Frequency found to be:');
092         (k+20791)*100e6/64e3
093         pause(10)
094     end
095     if (length(k) > 1)
096         k = median(k);
097         disp('oops')
098     end
099
100    est_data(i,k) = est_data(i,k) + 1; % Increment bin
101
102 end
103
104 est_data(1,:) = est_data(1,:).* alpha; % Sponge data to look like a larger estimate
105
106 clear x
107 clear index;
108 close(h);
109
110
111 % Load SNR1_F3 Data Matrix (65536x12) and Combination Matrix (495x8)
112 %***** ****
113 load c:\research\code\matlab\whites^2\snr1_f3
114 load c:\research\code\matlab\whites^2\cmb8_12
115 h = waitbar2(0,'Formulating SNR1 Estimation for f3...');
116
117 for n = 1:495
118
119     index = cmbo3;      % Load unique index pattern matrix.
120     waitbar2(n/495);
121
122     temp = x(:,index(n,:)); % Select current index pattern.

```

```

123 X = fft(temp);      % Perform 8 FFT's
124 X = X(lower_bound_index:upper_bound_index,:);    % Restrict to search bound
125 X_mag_squared = (X.*conj(X)).';
126 X_avg = (sum(X_mag_squared) ./ fft_avg).'; % Sum and average.
127
128 k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg)))); 
129 if insight
130     disp('Frequency found to be:')
131     (k+20791)*100e6/64e3
132     pause(10)
133 end
134 if (length(k) > 1)
135     k = median(k);
136     disp('oops')
137 end
138 est_data(2,k) = est_data(2,k) + 1; % Increment bin
139
140
141 end
142
143 est_data(2,:) = est_data(2,:).* alpha; % Sponge data to look like a larger estimate
144
145 clear x
146 clear index;
147 close(h);
148
149
150 % Load SNR2_F1 Data Matrix (65536x12) and Combination Matrix (495x8)
151 %***** ****
152 load c:\research\code\matlab\whites^2\snr2_f1
153 load c:\research\code\matlab\whites^2\cmb8_12
154 h = waitbar2(0,'Formulating SNR2 Estimation for f1...');
155
156 for n = 1:495
157
158     index = cmbo3;      % Load unique index pattern matrix.
159     waitbar2(n/495);
160
161     temp = x(:,index(n,:)); % Select current index pattern.
162     X = fft(temp);        % Perform 8 FFT's
163     X = X(lower_bound_index:upper_bound_index,:);    % Restrict to search bound
164     X_mag_squared = (X.*conj(X)).';
165     X_avg = (sum(X_mag_squared) ./ fft_avg).'; % Sum and average.
166
167     k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg)))); 
168     if insight
169         disp('Frequency found to be:')
170         (k+20791)*100e6/64e3
171         pause(10)
172     end
173     if (length(k) > 1)
174         k = median(k);
175         disp('oops')
176     end
177     est_data(3,k) = est_data(3,k) + 1; % Increment bin
178
179
180 end
181
182 est_data(3,:) = est_data(3,:).* alpha; % Sponge data to look like a larger estimate
183
184 clear x

```

```

185 clear index;
186 close(h);
187
188
189
190 % Load SNR2_F2 Data Matrix (65536x12) and Combination Matrix (495x8)
191 %*****%
192 load c:\research\code\matlab\whites^2\snr2_f2
193 load c:\research\code\matlab\whites^2\cmb8_12
194 h = waitbar2(0,'Formulating SNR2 Estimation for f2...');
195
196 for n = 1:495
197
198     index = cmbo3;      % Load unique index pattern matrix.
199     waitbar2(n/495);
200
201     temp = x(:,index(n,:)); % Select current index pattern.
202     X = fft(temp);        % Perform 8 FFT's
203     X = X(lower_bound_index:upper_bound_index,:);    % Restrict to search bound
204     X_mag_squared = (X.*conj(X)).';
205     X_avg = (sum(X_mag_squared) ./ fft_avg).'; % Sum and average.
206
207     k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg)))); 
208     if insight
209         disp('Frequency found to be:')
210         (k+20791)*100e6/64e3
211         pause(10)
212     end
213     if (length(k) > 1)
214         k = median(k);
215         disp('ooops')
216     end
217
218     est_data(4,k) = est_data(4,k) + 1; % Increment bin
219
220 end
221
222 est_data(4,:) = est_data(4,:).* alpha; % Sponge data to look like a larger estimate
223
224 clear x
225 clear index;
226 close(h);
227
228
229 % Load SNR2_F3 Data Matrix (65536x18) and Combination Matrix (1000x8)
230 %*****%
231 load c:\research\code\matlab\whites^2\snr2_f3
232 load c:\research\code\matlab\whites^2\mc_cmb2
233 h = waitbar2(0,'Formulating SNR2 Estimation for f3...');
234
235 for n = 1:1000
236
237     waitbar2(n/1000);
238
239     temp = x(:,index(n,:));
240     X = fft(temp);        % Perform 8 FFT's
241     X = X(lower_bound_index:upper_bound_index,:);
242     X_mag_squared = (X.*conj(X)).';
243     X_avg = (sum(X_mag_squared) ./ fft_avg).'; % Sum and average.
244
245     k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg)))); 
246     if (length(k) > 1)

```

```

247      k = median(k);
248      disp('oops')
249    end
250
251    est_data(5,k) = est_data(5,k) + 1; % Increment bin
252
253 end
254
255 clear x;
256 close(h);
257
258 % Load SNR2_F5 Data Matrix (65536x12) and Combination Matrix (495x8)
259 %*****%
260 load c:\research\code\matlab\whites^2\snr2_f5
261 load c:\research\code\matlab\whites^2\cmb8_12
262 h = waitbar2(0,'Formulating SNR2 Estimation for f5...');
263
264 for n = 1:495
265
266    index = cmbo3;      % Load unique index pattern matrix.
267    waitbar2(n/495);
268
269    temp = x(:,index(n,:)); % Select current index pattern.
270    X = fft(temp);        % Perform 8 FFT's
271    X = X(lower_bound_index:upper_bound_index,:); % Restrict to search bound
272    X_mag_squared = (X.*conj(X)).';
273    X_avg = (sum(X_mag_squared) ./ fft_avg).'; % Sum and average.
274
275    k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg)))); % Find the maximum value in the row
276    if insight
277        disp('Frequency found to be:')
278        (k+20791)*100e6/64e3
279        pause(10)
280    end
281    if (length(k) > 1)
282        k = median(k);
283        disp('oops')
284    end
285
286    est_data(6,k) = est_data(6,k) + 1; % Increment bin
287
288 end
289
290 est_data(6,:) = est_data(6,:).* alpha; % Sponge data to look like a larger estimate
291
292 clear x
293 clear index;
294 close(h);
295
296 % Load SNR3_F2 Data Matrix (65536x12) and Combination Matrix (495x8)
297 %*****%
298 load c:\research\code\matlab\whites^2\snr3_f2
299 load c:\research\code\matlab\whites^2\cmb8_12
300 h = waitbar2(0,'Formulating SNR3 Estimation for f2...');
301
302 for n = 1:495
303
304    index = cmbo3;      % Load unique index pattern matrix.
305    waitbar2(n/495);
306
307    temp = x(:,index(n,:)); % Select current index pattern.
308    X = fft(temp);        % Perform 8 FFT's

```

```

309 X = X(lower_bound_index:upper_bound_index,:); % Restrict to search bound
310 X_mag_squared = (X.*conj(X)).';
311 X_avg = (sum(X_mag_squared) ./ fft_avg).'; % Sum and average.
312
313 k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg)))); % Find the index of the maximum value
314 if insight
315     disp('Frequency found to be:')
316     (k+20791)*100e6/64e3
317     pause(10)
318 end
319 if (length(k) > 1)
320     k = median(k);
321     disp('oops')
322 end
323
324 est_data(7,k) = est_data(7,k) + 1; % Increment bin
325
326 end
327
328 est_data(7,:) = est_data(7,:).* alpha; % Sponge data to look like a larger estimate
329
330 clear x
331 clear index;
332 close(h);
333
334
335 mlver = version;
336 if (mlver(1) == 5) % Running on MATLAB 5.x
337     save -v4 dama_cap Fs Fc snr est_tolerance N number_of_estimates ...
338         est_data lower_bound_index upper_bound_index
339 else % Running on MATLAB 4.x
340     save dama_cap Fs Fc snr est_tolerance N number_of_estimates ...
341         est_data lower_bound_index upper_bound_index
342 end

```

ex1.m

```
001 %*****  
002 %  
003 % Experiment One: Finding Probabilities and MSE estimates of estimating  
004 % a complex sinusoid in noise.  
005 %  
006 % Purpose: The purpose of this code is to determine and prove the  
007 % relationship between the DAMA curves and actual MSE of  
008 % estimation. This baseline test will provide insight into the  
009 % simpler case of a complex sinusoid in noise which will then  
010 % be extended to the more complex DAMA carrier case.  
011 %  
012 % Programmer: Brad Scaife  
013 % Date: 2/14/99  
014 % Revision Date: 4/20/99  
015 % Current Revision: 1.10  
016 % Revision History:  
017 % 1.0 - Baseline  
018 % 1.01 - Corrected Noise Power and theoretical curve  
019 % 1.02 - Fixed indexing problem and clearing of mse  
020 % through each iteration.  
021 % 1.03 - Added support for estimation range compared  
022 % to some baseline DFT resolution for comparison.  
023 % 1.10 - Increasing the computational resolution of the  
024 % FFT so that the sim curve can more closely  
025 % match the Porat Curve.  
026 %  
027 % Notes: See Porat. As per indicated in Porat, the results of  
028 % estimation are valid only when the "rule of thumb" are  
029 % satisfied. Thus any processing of snr's below the ROT are  
030 % not valid with the theoretical curve.  
031 %*****  
032 clear all  
033 clc  
034  
035 % Program Parameters:  
036 %*****  
037 N = 512; % - Normally set to 512.  
038 L = 8*N; % - Computational Resolution.  
039 Nbase = N; % - Basic DFT resolution.  
040 fs = 800e3; % - Sampling frequency.  
041 Ts = 1/fs; % - Sampling interval.  
042 f = 200e3 + fs/(8*N); % - Sinusoid frequency.  
043 stime = 0; etime = (N-1)/fs;  
044 phi = 0; % - Phase offset.  
045 A = 1; % - Amplitude of sinusoid.  
046 Jw = 1; % - Set as window function - see Porat.  
047 D = Nbase*Ts; % - As per Porat.  
048  
049  
050 snr = [-20:.5:30]';  
051 number_estimates = 1000;  
052  
053 % Calculated Program Parameters  
054 %*****  
055 est_range = floor(fs/L)/2; % - Frequency estimation range as a function of  
056 % basic DFT resolution.  
057 f_lo = f - est_range; % - Lower "accurate" estimation bound.  
058 f_hi = f + est_range; % - Upper "accurate" estimation bound.  
059 k_lo = ceil(f_lo*N/fs); % - Lower index bound.  
060 k_hi = floor(f_hi*N/fs); % - Upper index bound.
```

```

061
062 if N == Nbase
063     k_hi = k_lo;
064 end
065
066 res_factor = N/L;           % - Resolution Factor
067
068 % Preallocate
069 %*****
070 success = zeros(1,length(snr));
071 mse_calc = zeros(1,length(snr));
072 mse = zeros(1,length(snr));
073 rmse = zeros(1,length(snr));
074 rmse_calc = zeros(1,length(snr));
075 mse_sum = 0;
076 %err_sum = 0;
077
078 % Signal Generation, Finding of true frequency
079 %*****
080 s = csin_gen(f,phi,A,fs,stime,etime);    % Verified power = 1.
081 signal_power = cov(s);
082 S = fft(s);
083 S_mag = abs(S);
084 %true_ind = find(S_mag == max(S_mag))-1
085 %f_true = true_ind/N*fs
086 %pause
087 f_true = f;
088 % true_ind = floor(f_true*N/fs)
089
090 % Display Parameters
091 %*****
092 disp(sprintf('True Frequency: %10.5f Hz',f_true))
093 disp(sprintf('Base Points: %d', N))
094 disp(sprintf('DFT Points: %d',L))
095 disp(sprintf('Base DFT Resolution: %10.5f Hz', fs/N))
096 disp(sprintf('Calculation DFT Resolution: %10.5f Hz', fs/L))
097 disp(sprintf('Resolution Factor: %d', res_factor))
098 disp(sprintf('Estimation Range: %10.5f Hz', est_range))
099 disp(sprintf('Lower Index Bound: %d', k_lo))
100 disp(sprintf('Upper Index Bound: %d', k_hi))
101 disp('Press a key to continue...') pause
102
103 for n = 1:length(snr)
104     snr(n)           % Current SNR to be tested.
105     snr_mod = sqrt(cov(s)/(10^(snr(n)/10))); % Standard Deviation of noise
106     % No = Ts * sqrt(2) * (snr_mod)^2;          % Noise Power in W/Hz.
107     % No = Ts * (2 * snr_mod^2)^2;
108     No = Ts * snr_mod^2;
109     mse_calc(n) = (6*No*Jw)/((2*pi)^2 * A^2 * D^3);
110     % rmse_calc(n) = sqrt( 6*Jw*sqrt( cov(s)/(10^(snr(n)/10) ) )/(100*pi^2))/D;
111     % rmse_calc(n) = (1/D)*sqrt(6*Jw*PG/(100*pi^2));
112     randn('seed',0);
113
114     for k = 1:number_estimates
115         v = [randn(N,1) + j*randn(N,1)]*snr_mod/sqrt(2);
116
117         %cov(s)           % Remove me
118         %cov(v)           % Remove me
119         %10*log10(cov(s) / cov(v)) % Remove me
120         %pause            % Remove me
121         y = s + v;
122

```

```

123      Y = fft(y,L);
124      Y_mag = abs(Y);
125      %Y_mag = Y_mag(1:N/2 + 1);
126
127      max_ind = min(find(Y_mag == max(Y_mag))) - 1;
128      f_found = max_ind/L*fs;
129
130      mse_sum = mse_sum + (f_true - f_found)^2;
131      % err_sum = err_sum + abs(f_true - f_found);
132
133      if (max_ind >= k_lo & max_ind <= k_hi)
134          success(n) = success(n) +1;
135      end
136
137
138
139  end
140
141  % mse(n) = (err_sum / number_estimates)^2;
142  mse(n) = mse_sum / number_estimates;
143  rmse(n) = sqrt(mse(n));
144  % err_sum = 0;
145  mse_sum = 0;
146
147  success(n) = success(n)/number_estimates;
148 end
149
150 rmse_calc = sqrt(mse_calc);

```

ex2.m

```
001 %*****  
002 %  
003 % Experiment Two: Finding Probabilities and MSE estimates of estimating  
004 % a real sinusoid in real noise.  
005 %  
006 % Purpose: The purpose of this code is to determine and prove the  
007 % relationship between the DAMA curves and actual MSE of  
008 % estimation. This baseline test will provide insight into the  
009 % simpler case of a complex sinusoid in noise which will then  
010 % be extended to the more complex DAMA carrier case.  
011 %  
012 % Programmer: Brad Scaife  
013 % Date: 3/18/99  
014 % Revision Date: 3/18/99  
015 % Current Revision: 1.0  
016 % Revision History:  
017 % 1.0 - Baseline  
018 %  
019 % Notes: See Porat. As per indicated in Porat, the results of  
020 % estimation are valid only when the "rule of thumb" are  
021 % satisfied. Thus any processing of snr's below the ROT are  
022 % not valid with the theoretical curve.  
023 %*****  
024 clear all clc  
025  
026 % Program Parameters:  
027 %*****  
028 N = 65536; % - Normally set to 512.  
029 Nbase = 65536; % - Basic DFT resolution.  
030 fs = 800e3; % - Sampling frequency.  
031 Ts = 1/fs; % - Sampling interval.  
032 f = 200e3 + fs/(4*N); % - Sinusoid frequency.  
033 stime = 0; etime = (N-1)/fs;  
034 phi = 0; % - Phase offset.  
035 A = 1; % - Amplitude of sinusoid.  
036 Jw = 1; % - Set as window function - see Porat.  
037 D = Nbase*Ts; % - As per Porat.  
038  
039  
040 snr = [-20:.2:10]';  
041 number_estimates = 100;  
042  
043 % Calculated Program Parameters  
044 %*****  
045 est_range = floor(fs/Nbase)/2; % - Frequency estimation range as a function of  
046 % basic DFT resolution.  
047 f_lo = f - est_range; % - Lower "accurate" estimation bound.  
048 f_hi = f + est_range; % - Upper "accurate" estimation bound.  
049 k_lo = ceil(f_lo*N/fs); % - Lower index bound.  
050 k_hi = floor(f_hi*N/fs); % - Upper index bound.  
051  
052 if N == Nbase  
053 k_hi = k_lo;  
054 end  
055  
056 res_factor = N/Nbase; % - Resolution Factor  
057  
058 % Preallocate  
059 %*****  
060 success = zeros(1,length(snr));
```

```

061 mse_calc = zeros(1,length(snr));
062 mse = zeros(1,length(snr));
063 rmse = zeros(1,length(snr));
064 rmse_calc = zeros(1,length(snr));
065 mse_sum = 0;
066 %err_sum = 0;
067
068 % Signal Generation, Finding of true frequency
069 %*****%
070 s = sinu_gen(f,phi,A,fs,stime,etime); % Verified power = 1.
071 signal_power = cov(s);
072 S = fft(s);
073 S_mag = abs(S);
074 %true_ind = find(S_mag == max(S_mag))-1
075 %f_true = true_ind/N*fs
076 %pause
077 f_true = f;
078 % true_ind = floor(f_true*N/fs)
079
080 % Display Parameters
081 %*****%
082 disp(sprintf('True Frequency: %10.5f Hz',f_true))
083 disp(sprintf('Base Points: %d', Nbase))
084 disp(sprintf('DFT Points: %d',N))
085 disp(sprintf('Base DFT Resolution: %10.5f Hz', fs/Nbase))
086 disp(sprintf('Calculation DFT Resolution: %10.5f Hz', fs/N))
087 disp(sprintf('Resolution Factor: %d', res_factor))
088 disp(sprintf('Estimation Range: %10.5f Hz', est_range))
089 disp(sprintf('Lower Index Bound: %d', k_lo))
090 disp(sprintf('Upper Index Bound: %d', k_hi))
091 disp('Press a key to continue...') pause
092
093 for n = 1:length(snr)
094     snr(n) % Current SNR to be tested.
095     snr_mod = sqrt(cov(s)/(10^(snr(n)/10))); % Standard Deviation of noise
096     % No = Ts * sqrt(2) * (snr_mod)^2; % Noise Power in W/Hz.
097     % No = Ts * (2 * snr_mod^2)^2;
098     No = Ts * snr_mod^2;
099     mse_calc(n) = (24*No*Jw)/((2*pi)^2 * A^2 * D^3);
100     % rmse_calc(n) = sqrt( 6*Jw*sqrt( cov(s)/(10^(snr(n)/10) ) )/(100*pi^2))/D;
101     % rmse_calc(n) = (1/D)*sqrt(6*Jw*PG/(100*pi^2));
102
103     for k = 1:number_estimates
104         v = randn(N,1) * snr_mod;
105
106         % cov(s) % Remove me
107         % cov(v) % Remove me
108         % 10*log10(cov(s) / cov(v)) % Remove me
109         % pause % Remove me
110         y = s + v;
111
112         Y = fft(y);
113         Y_mag = abs(Y);
114         Y_mag_p = Y_mag(1:N/2 + 1);
115
116         max_ind = min(find(Y_mag_p == max(Y_mag_p))) - 1;
117         f_found = max_ind/N*fs;
118
119         mse_sum = mse_sum + (f_true - f_found)^2;
120         % err_sum = err_sum + abs(f_true - f_found);
121
122         if (max_ind >= k_lo & max_ind <= k_hi)

```

```
123     success(n) = success(n) +1;
124 end
125
126
127
128 end
129
130 % mse(n) = (err_sum / number_estimates)^2;
131 mse(n) = mse_sum / number_estimates;
132 rmse(n) = sqrt(mse(n));
133 % err_sum = 0;
134 mse_sum = 0;
135
136 success(n) = success(n)/number_estimates;
137 end
138
139 rmse_calc = sqrt(mse_calc);
```

ex3.m

```
001 %*****  
002 %  
003 % Experiment Three: Finding Probabilities and MSE estimates of estimating  
004 % a BPSK carrier in real noise.  
005 %  
006 % Purpose: The purpose of this code is to determine and prove the  
007 % relationship between the DAMA curves and actual MSE of  
008 % estimation. This test is an extension of experiments one  
009 % and two where a BPSK carrier is under test.  
010 %  
011 % Programmer: Brad Scaife  
012 % Date: 3/30/99  
013 % Revision Date: 3/30/99  
014 % Current Revision: 1.0  
015 % Revision History:  
016 % 1.0 - Baseline  
017 %  
018 % Notes: See Porat. As per indicated in Porat, the results of  
019 % estimation are valid only when the "rule of thumb" are  
020 % satisfied. Thus any processing of snr's below the ROT are  
021 % not valid with the theoretical curve.  
022 %*****  
023 clear all clc  
024  
025 %-----  
026 % Program Parameters  
027 %-----  
028 N = 512;  
029 L = 2^0 * N;  
030 A = 1;  
031 Jw = 1;  
032 fft_avg = 25;  
033  
034  
035 %-----  
036 % Communication System Parameters  
037 %-----  
038 Fs = 800e3; % output signal sampling freq. (samples/s)  
039 Fc = 178e3; % BPSK carrier frequency in Hz. (cycles/sec)  
040 kc = floor(Fc*L/Fs);  
041 Rb = 10e3; % data rate (bits/s) Fs/Rb must be integer  
042 samples_per_bit = Fs/Rb; % must be integer  
043 D = N/Fs;  
044  
045 %-----  
046 % Simulation Parameters  
047 %-----  
048 number_of_estimates = 1000;% # of frequency estimates to perform for each SNR typically 10000  
049 snr = [-12:2:14]';  
050 msg = zeros(N,1);  
051 s = zeros(N,1);  
052 r = zeros(N,1);  
053  
054 %-----  
055 % Display Info  
056 %-----  
057 disp(sprintf('Sampling Frequency Fs: %5.15f',Fs));  
058 disp(sprintf('Carrier Frequency Fc: %5.15f',Fc));  
059 disp(sprintf('FFT Resolution (Data Supported): %5.15f',Fs/N));  
060 disp(sprintf('FFT Computational Resolution: %5.15f',Fs/L));
```

```

061
062
063 pause
064
065 %-----
066 % Begin Iterative SNR Loop
067 %-----
068
069 for k = 1:length(snr)
070
071     rand('seed',1000);
072     randn('seed',0);
073     mse_sum = 0;
074
075     noise_power = (A^2/2)/(10^(snr(k)/10));
076     No = (noise_power) / Fs;
077     help_factor = 1;
078     mse_calc(k) = (24*No*Jw)/(help_factor*(2*pi)^2 * (A^2/(4*Rb)) * D^3);
079
080     for l = 1:number_of_estimates
081
082         R_mag_sum = zeros(L/2,1);
083
084         for n = 1:fft_avg
085             %
086             % Generate BPSK Signal
087             %
088             msg = A*[cos(2*pi*([0:N-1])*Fc/Fs)].';
089             data = filter(ones(samples_per_bit,1),1,upsamp ...
090             ((-1).^(round(rand(ceil(N/samples_per_bit),1)))),...
091             samples_per_bit);
092
093             data = data(1:length(msg));
094             s = msg .* data;
095
096
097             noise = randn(length(msg),1) .* sqrt(cov(s)/(10^(snr(k)/10)));
098             r = s + noise;
099             r = r ./ sqrt(cov(r));
100
101
102
103             R = fft(r,L);
104             R_mag_sum = R_mag_sum + R(1:L/2).*conj(R(1:L/2));
105         end
106
107         R_mag = R_mag_sum ./ fft_avg;
108
109         kmax = min(find(R_mag == max(R_mag)));
110         f_est = (kmax-1)/L*Fs;
111
112         mse_sum = mse_sum + (Fc - f_est)^2;
113         %[Fc - f_est,kmax]
114
115     end
116
117     mse(k) = mse_sum/number_of_estimates;
118     rmse(k) = sqrt(mse(k));
119     rmse_calc(k) = sqrt(mse_calc(k));
120 end
121
122 clf

```

```
123 plot(snr,rmse_calc,'--')
124 hold on
125 plot(snr,rmse)
126 hold off
127 grid
```

B Motorola DSP 56303EVM Code

rev30.asm

```
001 ; REV 3.0
002
003 ; Just for convenience - delete later!!! Turns on/off D/A codec
004 ; bclr #19,x:M_CRBO ;Disable Rx on A-codec
005 ; bclr #18,x:M_CRBO ;Used to disable Tx interrupt
006
007
008     opt now
009
010 ; nolist
011     include 'ioequ.asm'
012     include 'intequ.asm'
013     include 'ada_equ.asm'
014     include 'vectors.asm'
015     include '7819equ.asm'
016 ;list
017
018 ;*****
019 ; Initial Layouts: This section of code sets up the D/A memory resources,
020 ; the program memory resources and defines the FFT macro.
021 ;*****
022     include 'CS4215.asm' ;D/A Memory Resources
023     include 'fftr2cn.asm' ;FFT Macro
024     include 'convnm.asm' ;Convolution Macro
025     include 'mlayout.dat' ;Memory Layout
026
027 ;*****
028 ; Fast Interrupt - IRQB
029 ;*****
030     org pli:I_IRQB
031     movep y:BB7819_DR,x:(r0) +
032     org pli:I_IRQB+1
033     bset #0,x:FLAGS
034
035     org p:$100
036 START
037 main
038
039 ;*****
040 ; Set Operating Frequency
041 ;*****
042     movep #CLK_RATE,x:M_PCTL ;Set PLL and Chip Operating Frequency
043
044 ;*****
045 ; Set Operating Parameters of DSP56303
046 ;*****
047     move #OP_MODE,omr ;Set Operating Mode of 303
048
049 ;*****
050 ; Setup Stack
051 ;*****
052     movec #0,sp ;clear hardware stack pointer
053     move #STACK,r6 ;initialise stack pointer
054     move #-1,m6 ;linear addressing
055
056
057
058 ;*****
```

```

059 ; Set AAR Wait States for External Memory(32k) and A/D Codec
060 ;*****
061     include 'ws_set.asm'      ;Set Wait States
062
063 ;*****
064 ; Set Up IRQB Interrupt Parameters.  IRQB is the interrupt designated to
065 ; the A/D Codec.
066 ;*****
067     ori    #$03,mr      ; Mask all interrupts until needed.
068     include 'core_ipl.asm' ; Set IRQB Interrupt Parameters
069
070
071 AAR2   equ    $fffc21      ; Compare Upper 12 bits to fffffx
072     movep #AAR2,x:M_AAR2 ; Setup AAR2
073
074
075     jsr    INIT          ; Register Initialization Routine
076     andi  #$fc,mr       ; Re-enable all interrupts
077
078 main_loop
079
080     jcld  #0,x:FLAGS,*  ;Wait for Sample In
081     bclr #0,x:FLAGS
082     jsr   process_sample
083     jmp   main_loop
084
085 process_sample
086     clr   a
087     move  x:SAMPLE_COUNTER,a0
088     dec   a
089     move  a0,x:SAMPLE_COUNTER
090     tst   a
091     jne   GET_NEXT_SAMPLE
092
093     ori   #$03,mr      ;Disable Interrupts
094
095     jsr   WIN_N_SCALE
096     jsr   COMPUTE_FFT
097     jsr   AVG_FFT
098
099     andi  #$fe,mr      ;Enables only 800k A/D
100     movep #IPRC,x:M_IPRC ;Re-enable A/D Codec
101
102 GET_NEXT_SAMPLE
103     rts
104
105 ; Subroutines:
106 ;*****
107     include 'comp_fft.asm'
108     include 'wsc.asm'
109     include 'avg_fft.asm'
110     include 'init.asm'
111     include 'get_bin.asm'
112     include 'sinvgid.asm'
113     include 'ada_init.asm'
114 echo
115 end

```

mlayout.dat

```
001 ;*****
002 ;mlayout.DAT: This data file is used with rev1.ASM to lay things out in memory
003 ;
004 ; Notes: For use with rev 2.1 code.
005 ;*****
006 ; References:
007 ;      DSP56300 Family Manual (300FM)
008 ;      DSP56303 User's Manual (303UM)
009 ;*****
010 ; Equates:
011 ;*****
012
013 ;*****
014 ; 56303 Processor Operating Parameters Control
015 ;*****
016 CLK_RATE      equ    $040004 ;Chip Operating Clock - See Section 9.3 300FM
017 OP_MODE       equ    $389   ;Chip Operating Mode
018               ;Please use either 389 or 3C9 for proper
019               ;operation. Please see DAMA Programming Notes
020               ;and 303UM:3-13 for details.
021 FS           equ    32000  ;Please set the same as D/A sample rate.
022 ;*****
023 ; DAMA Project Memory Settings. PLEASE DO NOT CHANGE!!! CODE WILL LIKELY NOT
024 ; FUNCTION. THE MEMORY HAS BEEN SPECIFICALLY SETUP UTILIZING ALL ON-CHIP
025 ; MEMORY.
026 ;*****
027 POINTS        equ    512    ;Number of Points (samples)
028 TABLE_SIZE    equ    512    ;Sine Wave Lookup Table Size (Will adjust output)
029 ITERS         equ    8      ;FFT Iterations
030 OFFSET        equ    128   ;Correction from Spectral Smearing due to Convolution.
031 ;OUTPUT_SEC    equ    2      ;Please enter duration of output in seconds.
032 ;OUT_LENGTH    equ    ECVI(FS*OUTPUT_SEC)
033
034 ; Long Memory:
035 ;*****
036     org l:$000a
037
038 SAMPLE_DATA   dsm    POINTS ;Signal buffer (0200 - 03ff)
039 FFT_DATA      dsm    POINTS ;FFT Output buffer (0400 - 05ff)
040 ;FFT_RESULT    dsm    POINTS ;Result FFT Data (0600 - 07ff)
041 COEFF         dsm    POINTS ;Sine-Cosine "Twiddle" Factor Lookup (0800 - 09ff)
042
043 ; X Memory:
044 ;*****
045     org x:$000a ;see ADA_INIT.ASM for why we start at x:$000a
046
047 SA_DATA_PTR   ds     1     ;SAMPLE_DATA Pointer Storage
048 FT_DATA_PTR   ds     1     ;FFT_DATA Pointer Storage
049 IFFT_PTR      ds     1     ;Imaginary FFT Data Pointer Storage
050 IFFT_MOD      ds     1     ;Imaginary FFT Data mod Storage
051 ;FT_RES_PTR    ds     1     ;FFT_RESULT Pointer Storage
052 MAG_PTR       ds     1     ;Magnitude Squared Data Pointer Storage
053 COEFF_PTR     ds     1     ;Coeff Pointer Storage
054 WAV_PTR       ds     1     ;Sine Wave Table Pointer Storage
055 WIN_PTR       ds     1     ;Window Pointer Storage
056 WIN_MOD       ds     1     ;Window Modulo Storage
057 SMF_PTR       ds     1     ;SMF Pointer Storage
058 CNVO_PTR      ds     1     ;Convolution Result Buffer Pointer Storage
059 FFT_COUNTER   ds     1     ;FFT Counter
060 SAMPLE_COUNTER ds     1     ;Sample Counter
```

```

061 MAX_VAL      ds    1 ;Maximum value storage
062 MAX_LOCATION ds    1 ;Holds address of max location
063 INT_DELTA    ds    1 ;Delta for Carrier Reconstruction.
064 FRAC_DELTA   ds    1 ;
065 R0_STORE     ds    1 ;r0 storage
066 R1_STORE     ds    1 ;r1 storage
067 R2_STORE     ds    1 ;r2 storage
068 R3_STORE     ds    1 ;r3 storage
069 R4_STORE     ds    1 ;r4 storage
070 R5_STORE     ds    1 ;r5 storage
071 R7_STORE     ds    1 ;r7 storage
072 M0_STORE     ds    1 ;m0 storage
073 M1_STORE     ds    1 ;m1 storage
074 M2_STORE     ds    1 ;m2 storage
075 M3_STORE     ds    1 ;m3 storage
076 M4_STORE     ds    1 ;m4 storage
077 M5_STORE     ds    1 ;m5 storage
078 M7_STORE     ds    1 ;m7 storage
079 N5_STORE     ds    1 ;n5 storage
080 N7_STORE     ds    1 ;n7 storage
081 FLAGS        ds    1 ;User Defined Flag Register
082 OUT_COUNTER  ds    1 ;Output Sample Counter
083 CNV_MEM      ds    1 ;For use in convolutional code.
084 STACK        equ   * ;Beginning of Stack
085
086     org x:$800
087 ;*****
088 ; Magnitude Squared Data
089 ;*****
090 MAG_SQ_DATA   dsm   POINTS/2
091
092     org x:$A00
093 ;*****
094 ; Generate Sine Wave Lookup Table
095 ;*****
096 TAB           dsm   TABLE_SIZE
097     include 'sintab.asm'
098     sintab TABLE_SIZE,TAB
099
100 ; Y Memory:
101 ;*****
102     org y:$0
103 ;*****
104 ; Generate Hamming Window w/ Prescale
105 ;*****
106 HAMM          dsm   POINTS      ;Hamming Window table.
107     include 'hamming.asm'
108     hamming POINTS,HAMM
109
110
111 ; Build Twiddle factor lookup tables for FFT Routine
112 ;*****
113     include 'sincos.asm'    ;Twiddle factor macro - builds lookup tables
114     sincos POINTS,COEFF   ;Build lookup tables.
115
116
117 ; Spectral Matched Filter
118 ;*****
119 SMF           dsm   POINTS/2
120     org y:SMF
121     include 'smf20.dat'
122

```

```
123
124 ; Convolution Output
125 ;*****
126 CNV_OUT dsm 2*POINTS-i
```

WSC.ASM

```
01 ;*****  
02 ; win_n_scale Subroutine  
03 ;  
04 ; Purpose: The purpose of this subroutine is to scale the input data to avoid  
05 ; overflow problems. This will have the effect of lowering the overall values  
06 ; of the spectrum but will not alter the shape of the spectrum.  
07 ;  
08 ; In:      x:(r0) - Sample Buffer  
09 ;           y:(r4) - Hamming buffer  
10 ;  
11 ; OUT:     x:(r0) - Sample Buffer w/ Window, scale, and iteration  
12 ;           adjustments  
13 ;  
14 ; Alters:   a,b,r0,r3,r4,x0,x1,y0,y1  
15 ;  
16 ; Written By: Brad Scaife  
17 ; Date:      2/20/98  
18 ; Platform:  Motorola DSP56303  
19 ; Calls:     None  
20 ;  
21 ; This code is verified for use with version three code. See rev30.asm.  
22 ;*****  
23 WIN_N_SCALE  
24  
25  
26     move    x:(r0)+,x0 y:(r4)+,y0      ;Preload values.  
27     do      #POINTS/2-1,ND_SCALE          ;  
28     mpyr   x0,y0,a x:(r0)+,x1 y:(r4)+,y1  ;x'(x)*(w(n)*scale/# iterations  
29     mpyr   x1,y1,b x:(r0)+,x0 y:(r4)+,y0  ;Second Iteration  
30     move    a,x:(r3)+                  ;Store a into sample buffer  
31     move    b,x:(r3)+                  ;Store b into sample buffer  
32 ND_SCALE  
33     mpyr   x0,y0,a x:(r0)+,x1 y:(r4)+,y1  ;Loop clean up: Two mults and  
34     mpyr   x1,y1,b                  ;corresponding writes to memory  
35     move    a,x:(r3)+                  ;Counters back to top of  
36     move    b,x:(r3)+                  ;buffer  
37  
38     rts
```

comp_fft.asm

```
01 ;*****  
02 ; COMPUTE FFT Subroutine  
03 ;  
04 ; Purpose: The purpose of this subroutine is to compute the FFT of the input  
05 ; signal and store it in memory.  
06 ;  
07 ; In:      r0,r1,r2,r3,r4,r5,m0,m1,m2,m3,m4,m5  
08 ; Out:     x:(r1),r0,r1,r2,r3,r4,r5,m0,m1,m2,m3,m4,m5  
09 ; Alters:   Everything  
10 ;*****  
11 COMPUTE_FFT  
12     move    r0,x:R0_STORE  
13     move    r1,x:R1_STORE  
14     move    r2,x:R2_STORE  
15     move    r3,x:R3_STORE  
16     move    r4,x:R4_STORE  
17     move    r5,x:R5_STORE  
18     move    r7,x:R7_STORE  
19     move    m0,x:M0_STORE  
20     move    m1,x:M1_STORE  
21     move    m2,x:M2_STORE  
22     move    m3,x:M3_STORE  
23     move    m4,x:M4_STORE  
24     move    m5,x:M5_STORE  
25     move    m7,x:M7_STORE  
26     move    n5,x:N5_STORE  
27     move    n7,x:N7_STORE  
28  
29     fftr2cn POINTS,SAMPLE_DATA,FFT_DATA,COEFF  
30  
31     move    x:N7_STORE,n7  
32     move    x:N5_STORE,n5  
33     move    x:M7_STORE,m7  
34     move    x:M6_STORE,m5  
35     move    x:M4_STORE,m4  
36     move    x:M3_STORE,m3  
37     move    x:M2_STORE,m2  
38     move    x:M1_STORE,m1  
39     move    x:M0_STORE,m0  
40     move    x:R7_STORE,r7  
41     move    x:R5_STORE,r5  
42     move    x:R4_STORE,r4  
43     move    x:R3_STORE,r3  
44     move    x:R2_STORE,r2  
45     move    x:R1_STORE,r1  
46     move    x:R0_STORE,r0  
47     rts
```

avg_fft.asm

```
01 ; for use with rev 2.1 code only AVG_FFT
02
03     move  x:(r1)+,x0 y:(r7)+,y0
04     do    #(POINTS/2),END_TLOOP
05     mpy   x0,x0,a x:(r2),y1
06     macr  y0,y0,a
07     add   y1,a
08     move  x:(r1)+,x0 y:(r7)+,y0
09     move  a,x:(r2)-
10
11 END_TLOOP
12     clr   b
13     move  x:FFT_COUNTER,b0
14     dec   b
15     move  b0,x:FFT_COUNTER
16     tst   b
17     jseq  GET_MAX_BIN
18
19 ;*****
20 ; Prepare to perform next FFT iteration
21 ;*****
22     move  #POINTS,x1      ;Reload sample counter for next sample
23     move  x1,x:SAMPLE_COUNTER ;buffering.
24
25 ;     move  #FFT_RESULT,r2      ;re-Setup FFT Result ptr
26     move  #MAG_SQ_DATA,r2      ;re-Setup Mag Squared data ptr
27     move  #FFT_DATA,r1        ;Setup FFT Data ptr
28     move  r1,r7      ;Imag. Pointer to FFT Buffer
29
30     move  $$0,x0
31     do    #POINTS,CLR_DAT      ;Clear FFT Data buffer
32     move  x0,x:(r1)          ;Real
33     move  x0,y:(r1)+          ;Imaginary
34 CLR_DAT
35
36     do    #POINTS,CLR_SMP
37     move  x0,x:(r0)
38     move  x0,y:(r0)-
39 CLR_SMP
40     rts
```

get_bin.asm

```
01 ;*****  
02 ; GET_MAX_BIN Subroutine  
03 ;  
04 ; Purpose: The purpose of this subroutine is to determine the frequency bin  
05 ; that has the largest component and then to determine the delta for the  
06 ; sine wave generation routine.  
07 ;  
08 ; In:      x:(r2)  
09 ; Out:     b  
10 ; Alters:  b,x1,y1,r2  
11 ;  
12 ; Notes:  For use with rev 2.1 code only!!  
13 ;*****  
14 GET_MAX_BIN  
15  
16 ;*****  
17 ; Clean Up From GET_BIN Subroutine  
18 ;*****  
19     move    $POINTS,x1      ;Reload sample counter for next sample  
20     move    x1,x:SAMPLE_COUNTER ;buffering.  
21     move    #FFT_DATA,r1      ;re-Setup FFT Data ptr  
22     move    r1,r7            ;Imag. Pointer to FFT Buffer  
23 ;     move    #FFT_RESULT,r2      ;re-Setup FFT Result ptr  
24     move    #MAG_SQ_DATA,r2      ;re-Setup Mag Sq Data ptr  
25 ;*****  
26 ; Perform SMF Convolution  
27 ;*****  
28     move    r0,x:R0_STORE  
29     move    r1,x:R1_STORE  
30     move    r4,x:R4_STORE  
31     move    m0,x:M0_STORE  
32     move    m1,x:M1_STORE  
33     move    m4,x:M4_STORE  
34  
35     convm   POINTS/2-1,MAG_SQ_DATA,SMF,CNV_OUT,CNV_MEM  
36  
37     move    x:R0_STORE,r0  
38     move    x:R1_STORE,r1  
39     move    x:R4_STORE,r4  
40     move    x:M0_STORE,m0  
41     move    x:M1_STORE,m1  
42     move    x:M4_STORE,m4  
43  
44     move    #CNV_OUT,r7  
45     move    $POINTS/2-2,m7  
46     clr    b  
47     do     #POINTS-1,ND_MAX  
48     move    x:(r7),x1      ;Bin Comparison  
49     cmp    x1,b            ;b-x1  
50     jlt    NEW_MAX        ;b will always hold max  
51     jmp    DUMMY  
52  
53 NEW_MAX  
54     move    x1,b          ;Store New Max Location  
55     move    r7,x:MAX_LOCATION  
56 DUMMY  
57     move    (r7)+  
58     nop  
59 ND_MAX  
60     move    x:MAX_LOCATION,b      ;Subtract max location from base
```

```
123 ;  
124     NOP             ;- Reserved  
125     NOP  
126 ;  
127     jmp   *           ;- SCI Receive Data  
128     NOP  
129 ;  
130     jmp   *           ;- SCI Receive Data w/ Exception Status  
131     NOP  
132 ;  
133     jmp   *           ;- SCI Transmit Data  
134     NOP  
135 ;  
136     jmp   *           ;- SCI Idle Line  
137     NOP  
138 ;  
139     jmp   *           ;- SCI Timer  
140     NOP  
141 ;  
142     NOP             ;- Reserved  
143     NOP  
144 ;  
145     NOP             ;- Reserved  
146     NOP  
147 ;  
148     NOP             ;- Reserved  
149     NOP  
150 ;  
151 ;  
152     jmp   *           ; Host receive data full  
153     NOP  
154 ;  
155 ;  
156     jmp   *           ;- Host transmit data empty  
157     NOP  
158 ;  
159     jmp   *           ; Available for Host Command  
160     NOP  
161     jmp   *           ; Available for Host Command  
162     NOP  
163     jmp   *           ; Available for Host Command  
164     NOP  
165     jmp   *           ; Available for Host Command  
166     NOP  
167     jmp   *           ; Available for Host Command  
168     NOP  
169     jmp   *           ; Available for Host Command  
170     NOP  
171     jmp   *           ; Available for Host Command  
172     NOP  
173     jmp   *           ; Available for Host Command  
174     NOP  
175     jmp   *           ; Available for Host Command  
176     NOP  
177     jmp   *           ; Available for Host Command  
178     NOP  
179     jmp   *           ; Available for Host Command  
180     NOP  
181     jmp   *           ; Available for Host Command  
182     NOP  
183     jmp   *           ; Available for Host Command  
184     NOP
```

```
61 : move #FFT_RESULT-1,y1 ;location to get the actual index
62 move #MAG_SQ_DATA-1,y1 ;location to get the actual index
63 sub y1,b ;Equals index imax
64 move #OFFSET,y0 ;
65 sub y0,b ;
66 move b1,n5 ;
67 jsr SINWGID ;end program
68
69 move #>ITERS,x1 ;Init FFT Counter
70 move x1,x:FFT_COUNTER ;Reset FFt counter for next iteration.
71 rts
```

sinwgid.asm

```
01 ;*****  
02 ; SINWGID Subroutine  
03 ;  
04 ; Purpose: The purpose of this subroutine is to generate a tone at a frequency  
05 ; based upon the delta value passed in from GET_BIN.ASM.  
06 ;  
07 ; In:      b  
08 ; Out:     n/a  
09 ; Alters:  a,x1,r5,n5,y0,  
10 ;  
11 ; Notes:  
12 ;   For use with rev 2.1 code.  
13 ;*****  
14 SINWGID  
15  
16       movep  $$0,x:M_IPRC      ;Disable A/D  
17       andi    $$fc,mr      ;Enable all interrupts  
18  
19 ;*****  
20 ; Initialization: D/A Codec and Setup Control Words. Only initialize the  
21 ; first time.  
22 ;*****  
23 ;       jset   #1,x:FLAGS,send_loop ;Skip after initial pass  
24 ;       jsr    ada_init          ; initialize codec  
25  
26       move   #TONE_OUTPUT,y0      ;set up control words  
27       move   y0,x:TX_BUFF_BASE+2  
28       move   #TONE_INPUT,y0  
29       move   y0,x:TX_BUFF_BASE+3  
30  
31 ;       bset   #1,x:FLAGS      ;Set after initialization.  
32  
33  
34 send_loop  
35       jset   #2,x:M_SSISR0,* ;wait for frame sync to pass  
36       jclr   #2,x:M_SSISR0,* ;wait for frame sync  
37  
38       move   x:(r5)+n5,y0  
39  
40 ;       clr    a              ;Test for end of duration of  
41 ;       move   x:OUT_COUNTER,a0 ;samples out phase.  
42 ;       dec    a  
43 ;       move   a0,x:OUT_COUNTER  
44 ;       tst    a  
45 ;       jeq    restart  
46  
47       move   y0,x:TX_BUFF_BASE ;transmit left  
48       move   y0,x:TX_BUFF_BASE+1 ;transmit right  
49       jmp    send_loop  
50  
51 restart  
52 ;       move   #OUT_LENGTH,x1 ;ReSet output duration counter.  
53 ;       move   x1,x:OUT_COUNTER ;  
54  
55       rts
```

WS_Set.asm

```
01 ;*****
02 ; Wait State Parameter Settings Routine
03 ;
04 ; Purpose: The purpose of this routine is to set the Bus Control Register (BCR)
05 ; to the proper number of wait states required by each AAR device. In the
06 ; DAMA Project, the AAR devices are:
07 ;
08 ;      32k SRAM
09 ;      Burr-Brown Codec (Operating @ 800kHz for DAMA Project)
10 ;
11 ;*****
12 ;Wait State Settings:
13 ;*****
14 DEFAULT_WS    equ     $0f      ;default are wait states
15 SRAM_WS       equ     $0f      ;32KW SRAM
16 FLASH_WS     equ     $00      ;FLASH
17 PERIPH_WS    equ     $0f      ;A/D Peripheral board
18
19
20 AREA0         equ     SRAM_WS
21 AREA1         equ     FLASH_WS
22 AREA2         equ     PERIPH_WS
23 AREA3         equ     SRAM_WS
24
25 BBS           equ     $0      ;Bus State
26 BLH           equ     $0      ;Bus Lock Hold
27 BRH           equ     $0      ;Bus Request Hold
28
29
30 BCR           equ     (BBS<<21)+(BLH<<22)+(BRH<<23)+(DEFAULT_WS<<16&M_BDFW) \
31             +(AREA3<<13&M_BA3W)+(AREA2<<10&M_BA2W)+(AREA1<<5&M_BA1W)+(AREA0&M_BA0W)
32
33     movep  #BCR,x:M_BCR      ;Initialize Bus Control Register
```

CONVM.ASM

```
convm      macro  length,xcoefs,hcoefs,result,cnv_mem
convm      ident   1,0
*****
; Macro Name: CONVM.ASM
-----
; Purpose:  The purpose of this macro is to provide the convolution
;           of two sequences stored in memory.  The algorithm does
;           a nested structure to minimize the memory required.
;
; Programmer: Brad Scaife
; Initial Date: 2/20/99
; Current Rev: 1.0
; Curr. Rev. Date: 2/20/99
; Revision History:
;
;       1.0 - Baseline
;
;-----
; Legal Statement:
; This DSP56xxx macro may be freely used with out the permission
; of the author.  The author provides the code with the intent that
; it is not to be used where such use may endanger life and property.
; Use of this macro code releases the author from ANY litagation both
; past, present, and future from ANY and ALL such liability claims.
; Use of this code is expressly permitted at your own risk.
;
;-----
; Resources Used:
;
; Registers Used:
;     a,b,r0,r1,r4,n0,n4,m0,m1,m4,x0,x1,y1
;
; Notes:
; Please note that this revision of the code requires the two
; input sequences to be of equal length.
*****
K          equ    length
K_ALL      equ    2*K-1

        move   #xcoefs,r0
        move   #K-1,m0
        move   #hcoefs,r4
        move   #K-1,m4
        move   #result,r1
        move   #K_ALL,m1

; Begin Calculation

        move   #0,x0
        clr    b x0,x:CNV_MEM
        clr    a

        move   x:(r0),x0 y:(r4),y1
        do    #K_ALL/2+1,FIRST
        move   b0,x:CNV_MEM
        do    x:CNV_MEM,END_F
```

```

    mac      x0,y1,a x:(r0)-,x0 y:(r4)+,y1

END_F
    inc      b
    move    $hcoefs,r4
    macr   x0,y1,a
    move    b0,n0
    move    a,x:(r1)+
    move    $xcoefs,r0
    nop
    nop
    clr      a
    move    (r0)+n0
    move    x:(r0)-,x0 y:(r4)+,y1

FIRST
    dec      b
    dec      b
    move    #>1,x1
    move    x:(r0)-,x0 y:(r4)+,y1
    do      #K_ALL/2, LAST
    move    b0,x:CNV_MEM
    do      x:CNV_MEM,END_L
    mac      x0,y1,a x:(r0)-,x0 y:(r4)+,y1

END_L
    move    #hcoefs+1,r4
    move    x1,n4
    move    b0,x:(r6)+
    move    x1,b0
    inc      b
    macr   x0,y1,a
    move    b0,x1
    move    $xcoefs+K-1,r0
    move    x:-(r6),b0
    move    a,x:(r1)+
    dec      b
    clr      a
    move    (r4)+n4
    move    x:(r0)-,x0 y:(r4)+,y1

LAST
    endm

```

cs4215.asm

```
01 ;---Buffer for talking to the CS4215
02
03     org    x:0
04 RX_BUFF_BASE    equ    *
05 RX_data_1_2      ds     1 ;data time slot 1/2 for RX ISR
06 RX_data_3_4      ds     1 ;data time slot 3/4 for RX ISR
07 RX_data_5_6      ds     1 ;data time slot 5/6 for RX ISR
08 RX_data_7_8      ds     1 ;data time slot 7/8 for RX ISR
09
10 TX_BUFF_BASE    equ    *
11 TX_data_1_2      ds     1 ;data time slot 1/2 for TX ISR
12 TX_data_3_4      ds     1 ;data time slot 3/4 for TX ISR
13 TX_data_5_6      ds     1 ;data time slot 5/6 for TX ISR
14 TX_data_7_8      ds     1 ;data time slot 7/8 for TX ISR
15
16 RX_PTR          ds     1 ; Pointer for rx buffer
17 TX_PTR          ds     1 ; Pointer for tx buffer
18
19 TONE_OUTPUT     EQU    HEADPHONE_EN+LINEOUT_EN+(4*LEFT_ATTN)+(4*RIGHT_ATTN)
20 TONE_INPUT       EQU    MIC_IN_SELECT+(15*MONITOR_ATTN)
21 CTRL_WD_12       equ    NO_PREAMP+HI_PASS_FILT+SAMP_RATE_32+STEREO+DATA_16 ;CLB=0
22 CTRL_WD_34       equ    IMMED_3STATE+XTAL1_SELECT+BITS_64+CODEC_MASTER
23 CTRL_WD_56       equ    $000000
24 CTRL_WD_78       equ    $000000
```

init.asm

```
01 ;*****  
02 ; INIT Subroutine  
03 ;  
04 ; Purpose: The purpose of this subroutine is to initialize pointers to memory  
05 ; and clear out buffers.  
06 ;  
07 ; In:      none  
08 ; OUT:     none  
09 ; Alters:   r0,m0,r1,m1,r2,m2,r3,m3,r4,m4  
10 ;  
11 ; Notes: For use with rev 3.0 code.  
12 ;*****  
13 INIT  
14     move    #SAMPLE_DATA,r0      ;Setup Sample buffer ptr  
15     move    r0,r3                  ;Alternate Sample buffer ptr  
16     move    #POINTS-1,m0          ;Setup Sample buffer mod  
17     move    m0,m3                  ;Alternate Sample Buffer mod  
18  
19     move    #$0,x0  
20     do      #POINTS,CLEAR_SAMPLE ;Clear Sample Buffer  
21     move    x0,x:(r0)            ;Real  
22     move    x0,y:(r0)+           ;Imaginary  
23 CLEAR_SAMPLE  
24  
25     move    #FFT_DATA,r1        ;Setup FFT Data ptr  
26     move    r1,r7                  ;Imag. Pointer to FFT Buffer  
27     move    #POINTS-1,m1          ;Setup FFT Data mod  
28     move    m1,m7                  ;Imag. FFT Buffer mod  
29     do      #POINTS,CLEAR_DATA ;Clear FFT Data buffer  
30     move    x0,x:(r1)            ;Real  
31     move    x0,y:(r1)+           ;Imaginary  
32 CLEAR_DATA  
33  
34 ;     move    #FFT_RESULT,r2      ;Setup FFT Result ptr  
35     move    #MAG_SQ_DATA,r2      ;  
36     move    #POINTS/2-1,m2          ;Setup FFT Result mod  
37     do      #POINTS/2,CLEAR_MAGSQ ;Clear FFT Result buffer  
38     move    x0,x:(r2)            ;Real  
39     move    x0,y:(r2)+           ;Imaginary  
40 CLEAR_MAGSQ  
41  
42     move    r7,x:IFFT_PTR       ;Store IFFT for r7 reuse.  
43     move    m7,x:IFFT_MOD       ;  
44     move    #CNV_OUT,r7          ;Setup Convolution Output ptr  
45     move    #2*POINTS-2,m7          ;Setup Conv. Output mod.  
46     do      #2*POINTS-1,CLEAR_CNV ;Clear Conv. Output  
47     move    x0,y:(r7)+           ;  
48 CLEAR_CNV  
49     move    x:IFFT_PTR,r7  
50     move    x:IFFT_MOD,m7  
51  
52     move    #HAMM,r4              ;Setup Hamming ptr  
53     move    #POINTS-1,m4          ;Setup Hamming mod  
54     move    #TAB,r5                ;Sine Table  
55     move    #TABLE_SIZE-1,m5          ;Sine Table mod  
56  
57     move    #POINTS,x1            ;Initialize Sample Counter  
58     move    x1,x:SAMPLE_COUNTER  
59     move    #>ITERS,x1             ;Init FFT Counter  
60     move    x1,x:FFT_COUNTER
```

```
61      move   x0,x:FLAGS          ;Clear User Defined Flag Register
62      move   #OUT_LENGTH,x1    ;Set output duration counter.
63      move   x1,x:OUT_COUNTER ;
64      ;
65      ;
66      ;
67      rts
```

hamming.asm

```
01 hamming    macro  points,hamm_loc
02 hamming    ident   1,2
03
04 py          equ     3.141592654
05 FREQ_INC   equ     2.0*py/@cvf(points-1)      ;frequency increment
06 SCALE_SHIFT equ     @CVI(@log(@cvf(points))/@log(2.0)) ;Shifts to Produce 1/POINTS
07 ;ITERATIONS equ     2      ;Number of FFT Iterations to perform. There
08                   ;exists a limit before overflow.
09 ;SCALE_FAC   equ     @cvf(points)*@cvf(ITERATIONS) ;Scale Factor
10
11     org y:hamm_loc
12 N          set    0
13           dup    points
14           dc     (0.54-0.46*@cos(FREQ_INC*@cvf(N)))/@cvf(points/2)
15 N          set    N+1
16           endm
17           endm  ;end of hamming macro
```

core_ipl.asm

```
01 ;*****  
02 ; Core Interrupt Priority Configuration Routine:  
03 ;  
04 ; Purpose: The purpose of this routine is to set the core Interrupt  
05 ; priorities. For the DAMA Project, only IRQB need concern us presently.  
06 ; Thus, only bits 5 to 3 are relevant. The following table suggests the  
07 ; proper settings:  
08 ;  
09 ;     IBL2: 0 for level triggering, 1 for edge triggering (DAMA uses edge)  
10 ;     IBL1-0:  
11 ;             00 01 10 11  
12 ;             -----  
13 ;             Enabled   No Yes Yes Yes  
14 ;             Priority - 0 1 2  
15 ;  
16 ; For details see 303UM:D-17.  
17 ;  
18 ; Current Settings:  
19 ;     Currently IRQB is the only interrupt enabled and it has been set to  
20 ;     priority level 2 (highest) and negative edge triggering.  
21 ;  
22 ; Written By: Tim Bagget  
23 ; Adapted By: Brad Scaife  
24 ; Date: 3/22/98  
25 ;  
26 ; Notes: For use with rev 2.1 code.  
27 ;*****  
28 ; CORE Interrupt Priority and Configuration  
29 IBM      equ      $1      ;IRQB trigger (0 level, 1 neg edge)  
30 IBP      equ      $1      ;IRQB priority level 0, 1, or 2;  
31  
32 IBL      equ      (IBM<<2)+(IBP+1)&3  
33  
34 ;IPRC    equ      IBL<<M_IBL0&M_IBL ;Disabled for the time being.  
35 IPRC    equ      $000038  
36      movep #IPRC,x:M_IPRC      ;Initialize Interrupt Priority/Config
```

sintab.asm

```
01 sintab      macro  tasiz,tab_loc
02 sintab      ident   1,2
03
04 pie         equ    3.141592654
05 TAB_INC     equ    2.0*pie/@cvf(tasiz)
06
07 org         x:tab_loc
08 N          set    0
09 dup         tasiz
10 dc          @sin(TAB_INC*@cvf(N))/2.0
11 N          set    N+1
12 endm
13 endm
14           ;end sine table generation macro
```

7819equ.asm

```
1 BB_ADR      equ      $0          ;DIP Switch Address SW1 ($0 - $3f)
2 BB7819_DR   equ      $FFFF80+BB_ADR ; ADS7819 Data Register
```

fftr2cn.asm

```
001 ;
002 ; This program originally available on the Motorola DSP bulletin board.
003 ; It is provided under a DISCLAIMER OF WARRANTY available from
004 ; Motorola DSP Operation, 6501 Wm. Cannon Drive W., Austin, Tx., 78735.
005 ;
006 ; Radix 2, In-Place, Decimation-In-Time FFT (fast).
007 ;
008 ; Last Update 18 Aug 88 Version 1.0
009 ;
010 fftr2cn macro points,data,odata,coef
011 fftr2cn ident 1,0
012 ;
013 ; Radix 2 Decimation in Time In-Place Fast Fourier Transform Routine
014 ;
015 ; Complex input and output data
016 ;     Real data in X memory
017 ;     Imaginary data in Y memory
018 ; Normally ordered input data
019 ; Normally ordered output data
020 ;     Coefficient lookup table
021 ;     -Cosine values in X memory
022 ;     -Sine values in Y memory
023 ;
024 ; Macro Call - fftr2cn points,data,odata,coef
025 ;
026 ;     points    number of points (16-32768, power of 2)
027 ;     data      start of data buffer
028 ;     odata     start of output data buffer
029 ;     coef      start of sine/cosine table
030 ;
031 ; Alters Data ALU Registers
032 ;     x1      x0      y1      y0
033 ;     a2      a1      a0      a
034 ;     b2      b1      b0      b
035 ;
036 ; Alters Address Registers
037 ;     r0      n0      m0
038 ;     r1      n1      m1
039 ;             n2
040 ;
041 ;     r4      n4      m4
042 ;     r5      n5      m5
043 ;     r7      n7      m7
044 ;
045 ; Alters Program Control Registers
046 ;     pc      sr
047 ;
048 ; Uses 6 locations on System Stack
049 ;
050 ; Latest Revision - 18 Aug-88
051 ;
052     move #data,r0          ;initialize input pointer
053     move #points/4,n0        ;initialize input and output pointers offset
054     move n0,n4
055     move n0,n7          ;initialize coefficient offset
056     move #points-1,m0        ;initialize address modifiers
057     move m0,m1          ;for modulo addressing
058     move m0,m4
059     move m0,m5
060 ;
```

```

061 ; Do first and second Radix 2 FFT passes, combined as 4-point butterflies
062 ;
063     move      x:(r0)+n0,x0
064     tfr x0,a      x:(r0)+n0,y1
065
066     do n0,_twopass
067     tfr y1,b      x:(r0)+n0,y0
068     add y0,a      x:(r0),x1          ;ar+cr
069     add x1,b      r0,r4          ;br+dr
070     add a,b       (r0)+n0          ;ar'=(ar+cr)+(br+dr)
071     subl b,a      b,x:(r0)+n0      ;br'=(ar+cr)-(br+dr)
072     tfr x0,a      a,x0          y:(r0),b
073     sub y0,a      y:(r4)+n4,y0      ;ar-cr
074     sub y0,b      x0,x:(r0)          ;bi-di
075     add a,b       y:(r0)+n0,x0      ;cr'=(ar-cr)+(bi-di)
076     subl b,a      b,x:(r0)          ;dr'=(ar-cr)-(bi-di)
077     tfr x0,a      a,x0          y:(r4),b
078     add y0,a      y:(r0)+n0,y0      ;bi+di
079     add y0,b      x0,x:(r0)+n0      ;ai+ci
080     add b,a       y:(r0)+,x0          ;ai'=(ai+ci)+(bi+di)
081     subl a,b      a,y:(r4)+n4      ;bi'=(ai+ci)-(bi+di)
082     tfr x0,a      b,y:(r4)+n4
083     sub y0,a      x1,b          ;ai-ci
084     sub y1,b      x:(r0)+n0,x0      ;dr-br
085     add a,b       x:(r0)+n0,y1          ;ci'=(ai-ci)+(dr-br)
086     subl b,a      b,y:(r4)+n4      ;di'=(ai-ci)-(dr-br)
087     tfr x0,a      a,y:(r4)+
088 _twopass
089 ;
090 ; Perform all next FFT passes except last pass with triple nested DO loop
091 ;
092     move #points/8,n1      ;initialize butterflies per group
093     move #4,n2      ;initialize groups per pass
094     move #-1,m2      ;linear addressing for r2
095     move $0,m7      ;initialize C address modifier for
096                      ;reverse carry (bit-reversed) addressing
097
098     do #@cvf(@log(points)/@log(2)-2.5),_end_pass  ;-1 ??? example: 7 passes for 1024 pt. FFT
099     move #data,r0      ;initialize A input pointer
100    move r0,r1
101    move n1,r2
102    move r0,r4          ;initialize A output pointer
103    move (r1)+n1          ;initialize B input pointer
104    move r1,r5          ;initialize B output pointer
105    move $coef,r7          ;initialize C input pointer
106    lua (r2)+,n0          ;initialize pointer offsets
107    move n0,n4
108    move n0,n5
109    move (r2)-          ;butterfly loop count
110    move      x:(r1),x1      y:(r7),y0          ;x1=br,y0=wi,lookup -sine and -cosine values
111    move      x:(r7)+n7,x0      y:(r0),b          ;b=ai, x0=wr,update C pointer, preload data
112    mac x1,y0,b      y:(r1)+,y1          ;y1=bi,b=ai+br*wi,
113    macr -x0,y1,b      y:(r0),a          ;a=ai again,b=ai+br*wi-bi*wr=bi'
114
115     do n2,_end_grp
116     do r2,_end_bfy          ;loop BF-1 times and start Radix 2 DIT BF kernel
117     subl b,a      x:(r0),b      b,y:(r4)          ;b=ar,a=ai-br*wi+bi*wr=ai', PUT bi'
118     mac -x1,x0,b      x:(r0)+,a      a,y:(r5)          ;a=br,b=ar-br*wr, PUT ai'
119     macr -y1,y0,b      x:(r1),x1          ;b=ar-(br*wr+bi*wi)=br',x1=next br
120     subl b,a      b,x:(r4)+      y:(r0),b          ;b=nai,a=2*ar-ar+br*wr+bi*wi=ar', PUT br'
121     mac x1,y0,b      y:(r1)+,y1          ;y1=nbi,b=nai+nbr*wi
122     macr -x0,y1,b      a,x:(r5)+      y:(r0),a          ;a=nai,b=nai+nbr*wi-nbi*wr=nbi' PUT ar'

```

```

123 _end_bfy
124     move (r1)+n1
125     subl b,a      x:(r0),b      b,y:(r4)          ;points to first B in next group
126     mac -x1,x0,b x:(r0)+n0,a  a,y:(r5)          ;PUT last bi' in a group
127     macr -y1,y0,b x:(r1),x1    y:(r7),y0          ;PUT last ai' in a group and update A pointer
128     subl b,a      b,x:(r4)+n4 y:(r0),b          ;PUT last br' in a group and update A' pointer
129     mac x1,y0,b   x:(r7)+n7,x0  y:(r1)+,y1        ;update W pointer
130     macr -x0,y1,b a,x:(r5)+n5 y:(r0),a          ;PUT last ar' in a group and update B' pointer
131 _end_grp
132     move n1,b1
133     lsr b      n2,a1    ;divide butterflies per group by two
134     lsl a      b1,n1    ;multiply groups per pass by two
135     move a1,n2
136 _end_pass
137 ;
138 ; Do last FFT pass
139 ;
140     move #2,n0      ;initialize pointer offsets
141     move n0,n1
142     move #points/4,n4 ;output pointer A offset
143     move n4,n5      ;output pointer B offset
144     move #data,r0    ;initialize A input pointer
145     move #odata,r4    ;initialize A output pointer
146     move r4,r2      ;save A output pointer
147     lua (r0)+,r1    ;initialize B input pointer
148     lua (r2)+n2,r5    ;initialize B output pointer
149     move #0,m4      ;bit-reversed addressing for output ptr. A
150     move m4,m5      ;bit-reversed addressing for output ptr. B
151     move #coef,r7    ;initialize C input pointer
152     move (r5)-n5    ;predecrement output pointer
153     move      x:(r1),x1    y:(r7),y0 ;x1=br,y0=wi
154     move      x:(r5),a      y:(r0),b ;a=?,b=ai
155
156     do n2,_lastpass ;Radix 2 DIT butterfly kernel with one butterfly per group
157     mac x1,y0,b   x:(r7)+n7,x0  y:(r1)+n1,y1  ;b=ai+br*wi,x0=ur, y1=bi
158     macr -x0,y1,b a,x:(r5)+n5  y:(r0),a      ;b=ai+br*wi-bi*wr=bi',a=ai, PUT previous ar'
159     subl b,a      x:(r0),b      b,y:(r4)          ;a=ai',b=ax,          PUT bi'
160     mac -x1,x0,b x:(r0)+n0,a  a,y:(r5)          ;b=ar-br*wr,a=ar,          PUT ai'
161     macr -y1,y0,b x:(r1),x1    y:(r7),y0          ;b=br',x1=nbr,y0=nwi
162     subl b,a      b,x:(r4)+n4  y:(r0),b          ;a=ar',b=nai,          PUT br'
163 _lastpass
164     move      a,x:(r5)+n5      ;          PUT ar'
165     endm

```

sincos.asm

```
01 ;
02 ; This program originally available on the Motorola DSP bulletin board.
03 ; It is provided under a DISCLAIMER OF WARRANTY available from
04 ; Motorola DSP Operation, 6501 Wm. Cannon Drive W., Austin, Tx., 78735.
05 ;
06 ; Sine-Cosine Table Generator for FFTs.
07 ;
08 ; Last Update 25 Nov 86 Version 1.2
09 ;
10 sincos macro points,coef
11 sincos ident 1,2
12 ;
13 ;      sincos -      macro to generate sine and cosine coefficient
14 ;                      lookup tables for Decimation in Time FFT
15 ;                      twiddle factors.
16 ;
17 ;      points -      number of points (2 - 32768, power of 2)
18 ;      coef    -      base address of sine/cosine table
19 ;                      negative cosine value in X memory
20 ;                      negative sine value in Y memory
21 ;
22 ; Latest revision - 25-Nov-86
23 ;
24
25 pi      equ     3.141592654
26 freq    equ     2.0*pi/@cvf(points)
27
28         org     x:coef
29 count   set     0
30         dup     points/2
31         dc      -@cos(@cvf(count)*freq)
32 count   set     count+1
33         endm
34
35         org     y:coef
36 count   set     0
37         dup     points/2
38         dc      -@sin(@cvf(count)*freq)
39 count   set     count+1
40         endm
41
42         endm    ;end of sincos macro
```

ada_equ.asm

```
001      page    132,60
002
003 ;*****
004
005 ;  ADA_EQU.ASM
006
007 ; Initialization constants to facilitate initialization of the CS4215
008
009 ;
010
011 ; Copyright (c) MOTOROLA 1996
012
013 ;          Semiconductor Products Sector
014
015 ;          Digital Signal Processing Division
016
017 ;
018
019 ;*****
020
021 ;
022
023
024
025 NO_PREAMP      equ     $100000
026
027 LO_OUT_DRV      equ     $080000
028
029 HI_PASS_FILT      equ     $008000
030
031 SAMP_RATE_9      equ     $003800      ; 9.6 kHz sample rate
032
033 SAMP_RATE_48      equ     $003000      ; 48   kHz sample rate
034
035 SAMP_RATE_32      equ     $001800      ; 32   kHz sample rate
036
037 SAMP_RATE_27      equ     $001000
038
039 SAMP_RATE_16      equ     $000800
040
041 SAMP_RATE_8       equ     $000000
042
043 STEREO          equ     $000400
044
045 DATA_8LIN        equ     $200300
046
047 DATA_8A           equ     $200200
048
049 DATA_8U           equ     $200100
050
051 DATA_16           equ     $200000
052
053 IMMED_3STATE      equ     $800000
054
055 XTAL1_SELECT      equ     $100000      ; 24.576 MHz
056
057 XTAL2_SELECT      equ     $200000      ; 16.9344 MHz
058
059 BITS_64           equ     $000000
060
```

```

061 BITS_128      equ     $040000
062
063 BITS_256      equ     $080000
064
065 CODEC_MASTER   equ     $020000
066
067 CODEC_TX_OFF    equ     $010000
068
069
070
071 ;CTRL_WD_12    equ     NO_PREAMP+HI_PASS_FILT+SAMP_RATE_48+STEREO+DATA_16 ;CLB=0
072
073 ;CTRL_WD_34    equ     IMMED_3STATE+XTAL1_SELECT+BITS_64+CODEC_MASTER
074
075 ;CTRL_WD_56    equ     $000000
076
077 ;CTRL_WD_78    equ     $000000
078
079
080
081 HEADPHONE_EN   equ     $800000
082
083 LINEOUT_EN     equ     $400000
084
085 SPEAKER_EN     equ     $004000
086
087 MIC_IN_SELECT   equ     $100000
088
089 LEFT_ATTN      equ     $010000 ;63*LEFT_ATTN = -94.5 dB, 1.5 dB steps
090
091 RIGHT_ATTN     equ     $000100 ;63*RIGHT_ATTN = -94.5 dB, 1.5 dB steps
092
093 LEFT_GAIN       equ     $010000 ;15*LEFT_GAIN = 22.5 dB, 1.5 dB steps
094
095 RIGHT_GAIN      equ     $000100 ;15*RIGHT_GAIN = 22.5 dB, 1.5 dB steps
096
097 MONITOR_ATTN    equ     $001000 ;15*MONITOR_ATTN = mute, 6 dB steps
098
099 ;OUTPUT_SET      equ     HEADPHONE_EN+LINEOUT_EN+(LEFT_ATTN*4)
100
101 ;INPUT_SET       equ     MIC_IN_SELECT+(15*MONITOR_ATTN)+(RIGHT_ATTN*4)

```

vectors.asm

```
001 ;
002     page    132,60
003 ;*****
004 ;  VECTORS.ASM
005 ;  Vector table for the 56303
006 ;
007 ;  Copyright (c) MOTOROLA 1996
008 ;          Semiconductor Products Sector
009 ;          Digital Signal Processing Division
010 ;
011 ;*****
012 ;
013     ORG      P:0
014 ;
015 vectors JMP     START           ; Hardware RESET
016 ;
017     jmp     *
018     NOP             ; Stack Error
019 ;
020     jmp     *
021     NOP           ;-- Debug Request Interrupt
022 ;
023     jmp     *
024     NOP           ;-- Debug Request Interrupt
025 ;
026     jmp     *
027     NOP           ;-- Trap
028 ;
029     jmp     *
030     NOP           ;-- NMI
031 ;
032     NOP           ;-- Reserved
033     NOP
034 ;
035     NOP           ;-- Reserved
036     NOP
037 ;
038     jsr     main        ;-- IRQA
039 ;
040     jmp     *
041     NOP           ;-- IRQB
042 ;
043     jmp     *
044     NOP           ;-- IRQC
045 ;
046     jsr     echo        ;-- IRQD
047 ;
048     jmp     *
049     NOP           ;-- DMA Channel 0
050 ;
051     jmp     *
052     NOP           ;-- DMA Channel 1
053 ;
054     jmp     *
055     NOP           ;-- DMA Channel 2
056 ;
057     jmp     *
058     NOP           ;-- DMA Channel 3
059 ;
060     jmp     *
```

```

061      NOP          ;- DMA Channel 4
062 ;
063      jmp *        ;- DMA Channel 5
064      NOP
065 ;
066      jmp *        ;- Timer 0 Compare
067      NOP
068 ;
069      jmp *        ;- Timer 0 Overflow
070      NOP
071 ;
072      jmp *        ;- Timer 1 Compare
073      NOP
074 ;
075      jmp *        ;- Timer 1 Overflow
076      NOP
077 ;
078      jmp *        ;- Timer 2 Compare
079      NOP
080 ;
081      jmp *        ;- Timer 2 Overflow
082      NOP
083 ;
084      jsr ssi_rx_isr      ;- ESSIO Receive Data
085 ;
086      jsr ssi_rxe_isr      ;- ESSIO Receive Data w/ Exception Status
087 ;
088      jsr ssi_rxls_isr      ;- ESSIO Receive last slot
089 ;
090      jsr ssi_tx_isr      ;- ESSIO Transmit Data
091 ;
092      jsr ssi_txe_isr      ;- ESSIO Transmit Data w/ Exception Status
093 ;
094      jsr ssi_txls_isr      ;- ESSIO Transmit last slot
095 ;
096      NOP          ;- Reserved
097      NOP
098 ;
099      NOP          ;- Reserved
100      NOP
101 ;
102      jmp *        ;- ESSI1 Receive Data
103      NOP
104 ;
105      jmp *        ;- ESSI1 Receive Data w/ Exception Status
106      NOP
107 ;
108      jmp *        ;- ESSI1 Receive last slot
109      NOP
110 ;
111      jmp *        ;- ESSI1 Transmit Data
112      NOP
113 ;
114      jmp *        ;- ESSI1 Transmit Data w/ Exception Status
115      NOP
116 ;
117      jmp *        ;- ESSI1 Transmit last slot
118      NOP
119 ;
120      NOP          ;- Reserved
121      NOP
122      NOP

```

```
185     jmp    *  
186     NOP    ; Available for Host Command  
187     jmp    *  
188     NOP    ; Available for Host Command  
189     jmp    *  
190     NOP    ; Available for Host Command  
191     jmp    *  
192     NOP    ; Available for Host Command  
193     jmp    *  
194     NOP    ; Available for Host Command  
195     jmp    *  
196     NOP    ; Available for Host Command  
197     jmp    *  
198     NOP    ; Available for Host Command  
199     jmp    *  
200     NOP    ; Available for Host Command  
201     jmp    *  
202     NOP    ; Available for Host Command  
203     jmp    *  
204     NOP    ; Available for Host Command  
205     jmp    *  
206     NOP    ; Available for Host Command  
207     jmp    *  
208     NOP    ; Available for Host Command  
209     jmp    *  
210     NOP    ; Available for Host Command  
211     jmp    *  
212     NOP    ; Available for Host Command  
213     jmp    *  
214     NOP    ; Available for Host Command  
215     jmp    *  
216     NOP    ; Available for Host Command  
217     jmp    *  
218     NOP    ; Available for Host Command  
219     jmp    *  
220  
221     NOP    ; Available for Host Command  
222     jmp    *  
223     NOP    ; Available for Host Command  
224     jmp    *  
225     NOP    ; Available for Host Command  
226     jmp    *  
227     NOP    ; Available for Host Command  
228     jmp    *  
229     NOP    ; Available for Host Command  
230     jmp    *  
231     NOP    ; Available for Host Command  
232     jmp    *  
233     NOP    ; Available for Host Command  
234     jmp    *  
235     NOP    ; Available for Host Command  
236     jmp    *  
237     NOP    ; Available for Host Command  
238     jmp    *  
239     NOP    ; Available for Host Command  
240     jmp    *  
241     NOP    ; Available for Host Command  
242     jmp    *  
243     NOP    ; Available for Host Command  
244     jmp    *  
245     NOP    ; Available for Host Command  
246     jmp    *
```

```
247      NOP          ; Available for Host Command
248      jmp *        ; Available for Host Command
249      NOP          ; Available for Host Command
250      jmp *        ; Available for Host Command
251      NOP          ; Available for Host Command
252      jmp *        ; Available for Host Command
253      NOP          ; Available for Host Command
254      jmp *        ; Available for Host Command
255      NOP          ; Available for Host Command
256      jmp *        ; Available for Host Command
257      NOP          ; Available for Host Command
258      jmp *        ; Available for Host Command
259      NOP          ; Available for Host Command
260      jmp *        ; Available for Host Command
261      NOP          ; Available for Host Command
262      jmp *        ; Available for Host Command
263      NOP          ; Available for Host Command
264      jmp *        ; Available for Host Command
265      NOP          ; Available for Host Command
266      jmp *        ; Available for Host Command
267      NOP          ; Available for Host Command
268      jmp *        ; Available for Host Command
269      NOP          ; Available for Host Command
270      jmp *        ; Available for Host Command
271      NOP          ; Available for Host Command
272      jmp *        ; Available for Host Command
273      NOP          ; Available for Host Command
274      jmp *        ; Available for Host Command
275      NOP          ; Available for Host Command
276      jmp *        ; Available for Host Command
277      NOP          ; Available for Host Command
278      jmp *        ; Available for Host Command
279      NOP          ; Available for Host Command
280      jmp *        ; Available for Host Command
281      NOP          ; Available for Host Command
282      jmp *        ; Available for Host Command
283
284
285      NOP          ; Available for Host Command
286      jmp *        ; Available for Host Command
287      NOP          ; Available for Host Command
288      jmp *        ; Available for Host Command
289      NOP          ; Available for Host Command
290      jmp *        ; Available for Host Command
291      NOP          ; Available for Host Command
292      jmp *        ; Available for Host Command
293      NOP          ; Available for Host Command
294      jmp *        ; Available for Host Command
295      NOP          ; Available for Host Command
296      jmp *        ; Available for Host Command
297      NOP          ; Available for Host Command
298      jmp *        ; Available for Host Command
299      NOP          ; Available for Host Command
300      jmp *        ; Available for Host Command
301      NOP          ; Available for Host Command
302      jmp *        ; Available for Host Command
303      NOP          ; Available for Host Command
304      jmp *        ; Available for Host Command
305      NOP          ; Available for Host Command
306      jmp *        ; Available for Host Command
307      NOP          ; Available for Host Command
308      jmp *
```

```
309      NOP          ; Available for Host Command
310      jmp          *
311      NOP          ; Available for Host Command
312      jmp          *
313      NOP          ; Available for Host Command
314      jmp          *
315      NOP          ; Available for Host Command
316      jmp          *
317      NOP          ; Available for Host Command
318 ;
319 ;
```

intequ.asm

```
01 ;*****  
02 ;  
03 ;      EQUATES for ONYXE 56302 interrupts  
04 ;  
05 ;      Last update: June 11 1995  
06 ;  
07 ;*****  
08  
09     page    132,55,0,0,0  
10     opt mex  
11     intequ ident 1,0  
12  
13     if  @DEF(I_VEC)  
14     ;leave user definition as is.  
15     else  
16     I_VEC    equ $0  
17     endif  
18  
19 ;-----  
20 ; Non-Maskable interrupts  
21 ;-----  
22 I_RESET  EQU  I_VEC+$00 ; Hardware RESET  
23 I_STACK   EQU  I_VEC+$02 ; Stack Error  
24 I_ILL    EQU  I_VEC+$04 ; Illegal Instruction  
25 I_DBG    EQU  I_VEC+$06 ; Debug Request  
26 I_TRAP   EQU  I_VEC+$08 ; Trap  
27 I_NMI    EQU  I_VEC+$0A ; Non Maskable Interrupt  
28  
29 ;-----  
30 ; Interrupt Request Pins  
31 ;-----  
32 I IRQA   EQU  I_VEC+$10 ; IRQA  
33 I IRQB   EQU  I_VEC+$12 ; IRQB  
34 I IRQC   EQU  I_VEC+$14 ; IRQC  
35 I IRQD   EQU  I_VEC+$16 ; IRQD  
36  
37 ;-----  
38 ; DMA Interrupts  
39 ;-----  
40 I DMA0   EQU  I_VEC+$18 ; DMA Channel 0  
41 I DMA1   EQU  I_VEC+$1A ; DMA Channel 1  
42 I DMA2   EQU  I_VEC+$1C ; DMA Channel 2  
43 I DMA3   EQU  I_VEC+$1E ; DMA Channel 3  
44 I DMA4   EQU  I_VEC+$20 ; DMA Channel 4  
45 I DMA5   EQU  I_VEC+$22 ; DMA Channel 5  
46  
47 ;-----  
48 ; Timer Interrupts  
49 ;-----  
50 I TIMOC  EQU  I_VEC+$24 ; TIMER 0 compare  
51 I TIMOOF EQU  I_VEC+$26 ; TIMER 0 overflow  
52 I TIM1C  EQU  I_VEC+$28 ; TIMER 1 compare  
53 I TIM1OF EQU  I_VEC+$2A ; TIMER 1 overflow  
54 I TIM2C  EQU  I_VEC+$2C ; TIMER 2 compare  
55 I TIM2OF EQU  I_VEC+$2E ; TIMER 2 overflow  
56  
57 ;-----  
58 ; ESSI Interrupts  
59 ;-----  
60 I SIORD  EQU  I_VEC+$30 ; ESSIO Receive Data
```

```

61 I_SIORDE EQU I_VEC+$32 ; ESSIO Receive Data With Exception Status
62 I_SIORLS EQU I_VEC+$34 ; ESSIO Receive last slot
63 I_SIOTD EQU I_VEC+$36 ; ESSIO Transmit data
64 I_SIOTDE EQU I_VEC+$38 ; ESSIO Transmit Data With Exception Status
65 I_SIOTLS EQU I_VEC+$3A ; ESSIO Transmit last slot
66 I_SI1RD EQU I_VEC+$40 ; ESSI1 Receive Data
67 I_SI1RDE EQU I_VEC+$42 ; ESSI1 Receive Data With Exception Status
68 I_SI1RLS EQU I_VEC+$44 ; ESSI1 Receive last slot
69 I_SI1TD EQU I_VEC+$46 ; ESSI1 Transmit data
70 I_SI1TDE EQU I_VEC+$48 ; ESSI1 Transmit Data With Exception Status
71 I_SI1TLS EQU I_VEC+$4A ; ESSI1 Transmit last slot
72
73 ; -----
74 ; SCI Interrupts
75 ; -----
76 I_SCIRD EQU I_VEC+$50 ; SCI Receive Data
77 I_SCIRDE EQU I_VEC+$52 ; SCI Receive Data With Exception Status
78 I_SCITD EQU I_VEC+$54 ; SCI Transmit Data
79 I_SCIIL EQU I_VEC+$56 ; SCI Idle Line
80 I_SCITM EQU I_VEC+$58 ; SCI Timer
81
82 ; -----
83 ; HOST Interrupts
84 ; -----
85 I_HRDF EQU I_VEC+$60 ; Host Receive Data Full
86 I_HTDE EQU I_VEC+$62 ; Host Transmit Data Empty
87 I_HC EQU I_VEC+$64 ; Default Host Command
88
89 ; -----
90 ; INTERRUPT ENDING ADDRESS
91 ; -----
92 I_INTEND EQU I_VEC+$FF ; last address of interrupt vector space
93 LIST

```

sincos.asm

```
01 ;  
02 ; This program originally available on the Motorola DSP bulletin board.  
03 ; It is provided under a DISCLAIMER OF WARRANTY available from  
04 ; Motorola DSP Operation, 6501 Wm. Cannon Drive W., Austin, Tx., 78735.  
05 ;  
06 ; Sine-Cosine Table Generator for FFTs.  
07 ;  
08 ; Last Update 25 Nov 86 Version 1.2  
09 ;  
10 sincos macro points,coef  
11 sincos ident 1,2  
12 ;  
13 ;      sincos -      macro to generate sine and cosine coefficient  
14 ;                      lookup tables for Decimation in Time FFT  
15 ;                      twiddle factors.  
16 ;  
17 ;      points -      number of points (2 - 32768, power of 2)  
18 ;      coef -        base address of sine/cosine table  
19 ;                      negative cosine value in X memory  
20 ;                      negative sine value in Y memory  
21 ;  
22 ; Latest revision - 25-Nov-86  
23 ;  
24  
25 pi     equ    3.141592654  
26 freq   equ    2.0*pi/@cvf(points)  
27  
28      org    x:coef  
29 count  set    0  
30 dup    points/2  
31 dc     -@cos(@cvf(count)*freq)  
32 count  set    count+1  
33 endm  
34  
35      org    y:coef  
36 count  set    0  
37 dup    points/2  
38 dc     -@sin(@cvf(count)*freq)  
39 count  set    count+1  
40 endm  
41  
42      endm    ;end of sincos macro
```

ada_init.asm

```
001      page     132,60
002 ;*****ADA_INIT.ASM Ver.2.0*****
003 ;   Example program to initialize the CS4215
004 ;
005 ;
006 ;   Copyright (c) MOTOROLA 1995, 1996
007 ;           Semiconductor Products Sector
008 ;           Digital Signal Processing Division
009 ;
010 ;       History:
011 ;           14 June 1996: RLR/LJD - ver.1.0
012 ;*****PLEASE NOTE: For use with rev 2.1 code.*****
013 ;
014 ;*****initialize the CS4215 codec*****
015 ;*****PROGRAM OUTLINE:
016 ;
017 ;       portc usage:
018 ;       bit8: SSI TX (from DSP to Codec)
019 ;       bit7:
020 ;       bit6:
021 ;       bit5:
022 ;       bit4: codec reset (from DSP to Codec)
023 ;       bit3:
024 ;           bit2: data/control bar
025 ;               0=control
026 ;               1=data
027 ;
028 ;*****initialize ssi -- fsync and sclk == outputs
029 ;*****org p:
030 ;ada_init
031 ;
032 ;
033 ;       movep  $$0000,x:M_PCRC      ; turn off ESSIO port (for now)
034 ;       movep  $$103807,x:M_CRA0      ; 40MHz/16 = 2.5MHz SCLK, WL=16 bits, 4W/F
035 ;       movep  $$ff313C,x:M_CRB0      ; RIE,TIE,RLIE,TLIE,RE,TE,sc2/sck outputs
036 ;       movep  $$0003,x:M_PRRC      ; setup pd0 and pd1 as gpio output
037 ;       movep  $$0,x:M_PDRC      ; send out a 0 on DC~ and RST_CODEC~
038 ;
039 ;-----reset delay for codec -----
040 ;       do      $1000,_delay_loop
041 ;       rep     $2000          ; 100 us delay (assuming 40MHz VCO)
042 ;       nop
043 ;_delay_loop
044 ;
045 ;       org     p:
```

```

061      bset    #0,x:M_PDRC          ; sends out a 1 on pd0 (rst_codec=1)
062      movep   $$0008,x:M_IPRP        ; set interrupt priority level for ESSIO to 1
063      andi    $$FC,mr             ; enable interrupts
064
065 ;*****+
066 ; The following data sets up the CS4215 control mode data:
067 ;     (CTS = Control Time Slot, U/LN = upper/lower Nibble)
068 ;
069 ;     +----- CTS1-UN: 0 0 1 MLEB 0 0 0 0
070 ;     |+---- CTS1-LN: OLB CLB X X 0 0 0 0
071 ;     ||+--- CTS2-UN: HPF X DFR2 DFR1 0 0 1 0
072 ;     |||+-- CTS2-LN: DFRO ST DF1 DFO 1 1 0 0
073 ; x0 = $002Cxx
074 ;
075 ;     +----- CTS3-UN: ITS MCK2 MCK1 MCK0 1 0 0 0
076 ;     |+---- CTS3-LN: BSEL1 BSELO XCLK XEN 1 0 0 0
077 ;     ||+--- CTS4-UN: TEST TEST TEST TEST (TEST MUST BE 0)
078 ;     |||+-- CTS4-LN: TEST TEST ENL DAD 0 0 0 0
079 ; x0 = $8800xx
080 ;*****+
081
082 ;--- set up buffer with control mode data
083      move   #CTRL_WD_12,x0
084      move   x0,x:TX_BUFF_BASE
085      move   #CTRL_WD_34,x0
086      move   x0,x:TX_BUFF_BASE+1
087      move   #CTRL_WD_56,x0
088      move   x0,x:TX_BUFF_BASE+2
089      move   #CTRL_WD_78,x0
090      move   x0,x:TX_BUFF_BASE+3
091
092      movep  $$003C,x:M_PCRC       ;turn on ESSIO except for sc0 and sc2
093
094 ;
095 ; CLB == 0
096 ;
097      jclr   #3,x:M_SSISR0,*      ; wait until rx frame bit==1
098      jset   #3,x:M_SSISR0,*      ; wait until rx frame bit==0
099      jclr   #3,x:M_SSISR0,*      ; wait until rx frame bit==1
100      jset   #18,x:RX_BUFF_BASE,* ; loop until CLB set
101
102 ;
103 ; CLB == 1
104 ;
105      bset   #18,x:TX_BUFF_BASE  ;set CLB
106      do    #4,_init_loopB
107      jclr   #2,x:M_SSISR0,*      ; wait until tx frame bit==1
108      jset   #2,x:M_SSISR0,*      ; wait until tx frame bit==0
109 _init_loopB
110      movep  $$0000,x:M_PCRC       ; disable ESSIO
111
112 ;*****+
113 ; now CLB should be 1 -- re-program fsync and sclk direction (i/p) -- also,
114 ; circular buffer pointers for echoing data r0=current, r1=old data to send
115 ; 1 frame later
116 ;
117      movep  $$103807,x:M_CRA0      ; 40MHz/16 = 2.5MHz SCLK, WL=16 bits, 4W/F
118      movep  $$FF310C,x:M_CRB0      ; sckd and fsync (sc02) as inputs
119      movep  $$0003,x:M_PDRC        ; D/C pin = 1 ==> data mode
120      movep  $$003C,x:M_PCRC       ; turn on ESSIO except for sc0 and sc2
121      rts
122

```

```

123 ;*****
124 ;   SSIO_ISR.ASM    Ver.2.0
125 ;   Example program to handle interrupts through
126 ;       the 56303 SSIO to move audio through the CS4215
127 ;
128 ; Copyright (c) MOTOROLA 1995, 1996
129 ;             Semiconductor Products Sector
130 ;             Digital Signal Processing Division
131 ;
132 ; upon entry:
133 ;           R6 must be the stack pointer
134 ; corrupts:
135 ;           R6
136 ;
137 ; History:
138 ;     14 June 1996: RLR/LJD - ver 1.0
139 ;*****
140
141
142 ;----the actual interrupt service routines (ISRs) follow:
143
144 ;***** SSI TRANSMIT ISR *****
145 ssi_txe_isr
146     bclr    $4,x:M_SSISR0      ; Read SSISR to clear exception flag
147                           ; explicitly clears underrun flag
148 ssi_tx_isr
149     move    r0,x:(r6)+        ; Save r0 to the stack.
150     move    m0,x:(r6)+        ; Save m0 to the stack.
151     move    #3,m0            ; Modulus 4 buffer.
152     move    x:TX_PTR,r0      ; Load the pointer to the tx buffer.
153     nop
154     movep   x:(r0)+,x:M_TX00 ; SSI transfer data register.
155     move    r0,x:TX_PTR      ; Update tx buffer pointer.
156     move    x:-(r6),m0        ; Restore m0.
157     move    x:-(r6),r0        ; Restore r0.
158     rti
159
160 ;***** SSI TRANSMIT LAST SLOT ISR *****
161 ssi_txls_isr
162     move    r0,x:(r6)+        ; Save r0 to the stack.
163     move    $TX_BUFF_BASE,r0  ; Reset pointer.
164     move    r0,x:TX_PTR      ; Reset tx buffer pointer just in
165                           ; case it was corrupted.
166     move    x:-(r6),r0        ; Restore r0.
167     rti
168
169 ;***** SSI receive ISR *****
170 ssi_rxe_isr
171     bclr    $5,x:M_SSISR0      ; Read SSISR to clear exception flag
172                           ; explicitly clears overrun flag
173 ssi_rx_isr
174     move    r0,x:(r6)+        ; Save r0 to the stack.
175     move    m0,x:(r6)+        ; Save m0 to the stack.
176     move    #3,m0            ; Modulo 4 buffer.
177     move    x:RX_PTR,r0      ; Load the pointer to the rx buffer.
178     nop
179     movep   x:M_RX0,x:(r0)+  ; Read out received data to buffer.
180     move    r0,x:RX_PTR      ; Update rx buffer pointer.
181     move    x:-(r6),m0        ; Restore m0.
182     move    x:-(r6),r0        ; Restore r0.
183     rti
184

```

```
185 ;***** SSI receive last slot ISR *****
186 ssi_rxls_isr
187     move    r0,x:(r6)+          ; Save r0 to the stack.
188     move    #RX_BUFF_BASE,r0      ; Reset rx buffer pointer just in
189                               ; case it was corrupted.
190     move    r0,x:RX_PTR          ; Update rx buffer pointer.
191     move    x:-(r6),r0          ; Restore r0.
192     rti
```

References

- [1] Stephen Horan. An operational concept for a demand assignment multiple access system for the space network. Technical report, New Mexico State University, 1996.
- [2] Phillip L. De Leon. Real-time dsp-based carrier recovery with unknown doppler shift. Technical report, New Mexico State University, 1998.
- [3] Monica M. Sanchez. Doppler extraction for a demand assignment multiple access service for NASA's space network. Technical report, New Mexico State University, August 1996.
- [4] Boaz Porat. *A Course in Digital Signal Processing*. John Wiley and Sons, Inc., first edition, 1997.
- [5] NASA. Tdrss system description. world wide web, 1998.
- [6] Franklin Miller. *College Physics*. Harcourt Brace Jovanovich, Inc., fourth edition, 1977.
- [7] John G. Proakis. *Digital Communications*. McGraw-Hill, Inc., third edition, 1995.
- [8] Leon W. Couch II. *Digital and Analog Communication Systems*. Prentice-Hall, Inc., fifth edition, 1997.
- [9] Simon Haykin. *Adaptive Filter Theory*. Prentice-Hall, Inc., third edition, 1996.
- [10] Frank Hartman and Cliff Baxter. Private communication, July 1998.
- [11] Motorola. *DSP 36303 User's Manual*. Motorola, Inc., first edition, 1995.

- [12] Motorola. *DSP 56300 Family Manual*. Motorola, Inc., second edition, 1995.
- [13] Tim Bagget. High speed A/D interface for carrier doppler tracking. Master's thesis, New Mexico State University, 1998.
- [14] Sophocles J. Orfanidis. *Introduction to Signal Processing*. Prentice-Hall, Inc., first edition, 1996.

